Learning Programming

with your Children

8 steps to help you and your child learn and practice programming

Learning Programming with your Children

8 steps to help you and your child learn and practice programming

by
Nicos Ioannou

Editors: Emily Millett and Timothy Isaacs
Designer & Illustrator: Andreas Kyprianou - www.DBspectrum.com

Copyright © August 2016 by Nicos Ioannou Nicosia, Cyprus
Second edition August 2019
https://www.patreon.com/namatreasure
(https://vimeo.com/namatreasure/videos)
namatreasurebook@gmail.com

All Rights Reserved
No part of this book may be reproduced in any form, either by photocopying or by electronic or mechanical means, including information storage or retrieval systems, without permission in writing from both the copyright owner and the publisher of this book. This book is not intended to give any legal, medical and/or financial advice to its readers.

Dedicated to you the reader, the parent, seeking the best and most useful gift for your children, seeking something that can bring you closer, that allows you to engage in fun and educational activities together.

Dedicated to you the reader, who has realized that conventional education is not as fun and engaging as it should be and does not always attract or maintain children's attention.

I have been in your position and this book is the fruit of my efforts to teach my nephews, children, their friends and any other child who might be interested in programming – but teach them in a playful and fun way.

This book is dedicated to you because it has been made for you. It is also dedicated to the Cyprus-America Scholarship Program (CASP) and the Fulbright Program.

Table of Contents

Learning Programming with your Children 5

Motivation 6

Ch.How to use this Book 10

Ch.1 Introduction to Programming as a New Language 13

Ch.2 Namatreasure Programming Environment 22

Ch.3 Computer Programming Environments - Scratch and Blockly 39

Ch.4 Tips to make an environment interesting 52

Ch.5 Competitions are a powerful driver. 68

Ch.6 Storytelling 75

Ch.7 Analysis before execution 88

Ch.8 Teams produce more work 97

Ch.9 To 3D world and beyond 100

Appendix 1. Level and Certificates 110

Appendix 2 Programming instruction 116

Appendix 3 Further Reading 125

Acknowledgments 129

About Nicos Ioannou 130

Learning Programming with your Children

8 steps to help you and your child learn and practice programming

Motivation

Before you read any book, you might wonder whether it will be important for you. This book is about you and your children and how together, you can learn to communicate with, and control the billions of devices which surround us today. This book is based on 20 years of experience and a never-ending effort to guide children towards learning programming (coding) through games and play. Ever since computers first appeared, something called a ‘programming language’ was invented. What is this programming language? Put simply, it is the language that we speak to these computerized devices and machines. And as these computerized machines become more and more intelligent, they occupy more and more of our world, both in the workplace and at home.

- Computers, mobile phones, tablets, smart TVs, smart polymixers, smart ovens and smart refrigerators exist in the market and in our homes
- Autonomous vehicles, smart houses, smart offices and Artificial Intelligence in the shape of robot-maids or e-secretaries will soon be available on the market
- We hold, see, use and listen to intelligent devices: they are our new pets and helpers
- A swarm of intelligent devices will always be around us
- Program them to serve you. If you do not know their program, they may spy on you

For these reasons, it is important for us to talk to these digital devices. If we don’t talk to them, someone else will. What is the importance of programming? Could we go as far as to say that programming is like a new alphabet?

The only similarity between programming and the alphabet, is that in the past, we would describe someone as educated if they could read and write. Today when we talk about a technology-educated person

we mean someone who, amongst other things, can program and solve problems using programming.

Learning programming is like learning another language; but an easy language, like a subset of your native tongue. Just like mathematics, programming is a universal language, and the similarities don't end there. Programming is also like mathematics because it is a discipline which children must begin learning from an early age and continue learning for the rest of their lives. We use it both implicitly and explicitly in all aspects of our everyday lives. Programming is a subset of our spoken language and it is easy to learn. Anyone can learn programming and through programming, can then subsequently learn to make mobile applications, internet games or learn to program home automation devices. If you stay with us until the end and practice the examples, you will gain a very good understanding of these graphical programming languages and above all, you will master the programming principles so that you can easily pick up any programming language.

Another way of looking at this subject, is to look at the important people who program. Most of the biggest companies that influence our lives are involved in programming. For example, Apple, Autodesk, Adobe, Amazon, e-bay, Dropbox and Google all have products which are created using programming. The people who founded or run these companies were once programmers or engineers who made computers, or entrepreneurs who realized the importance of programming. Therefore, it is a very important discipline in our lives and people that have managed to master this have gone a long way.

But, what can we do with programming? We can use it in everything, literally everything from building web pages, programming

appliances, garden watering systems and soon we will be programming our energy management systems and ventilation systems (for example opening and closing the windows and doors). In the near future, we will even program our cars. In the work area we program control systems, the e-shop and automation systems. Coming soon is the electronic office and, of course, your new e-secretary; and the list goes on and on. Programming has changed health support systems, factory floors and the design of new medicines. Programming is making fun games, movies, and is also changing education as it has changed the way we communicate. So

programming is everywhere, and it is important. Even artists program to make animation or art in motion as it is also known.

Learning a new language must start at a very early age and we have to practice very often as we do with mathematics. The more we practice, the more we learn and the more recipes (in other books this is also called algorithms) get stored in our minds. Whenever we need to solve a problem using a program, we can draw on our stored recipes and combine them to create something new.

“Seeking the Namatreasure” is an educational tabletop game and the predecessor of this book. Both the educational game and the book have one purpose: to awaken parents so that they understand the importance of learning programming. If the school does not provide programming lessons, courses or workshops, the parents should take the responsibility and do it themselves. This book aims to be a backpack, with tools that will help parents on this journey of learning together with their children. Most parents do not know programming themselves, so they will have to learn it first in order to teach their children. Please believe us, it is not difficult, and it comes naturally to all humans. It is not difficult because there are tools that are there to help you and we will cover them in this book.

KEY CONCEPTS: Programming is fun. You can also think of people as intelligent devices that are programmed (they act upon a predetermined pattern of behavior). Programming computers is done my mimicking the way we humans think and learn. Soon, by reading this book you will learn how to create a world of robots or characters that will interact with each other and you, following the instructions that you have implanted into them. It is obvious that robots (machines) are not naturally intelligent, but they may behave in a way that intelligent beings do, because they can store and then execute intelligent behavior. With programming you can solve many real problems. You can even program wearable devices such as LED lights on costumes. Computers do not think of themselves, but they can execute complex programs that make them behave intelligently.

How to use this Book

If you want to teach your children programming, there is a vast amount of information, tutorials and courses available on the internet, both free and paid. The process of learning through these online courses can be long and tedious and you may get lost, or you may not have the time. This book is a shortcut that helps you reach the destination quickly and safely, while also having the opportunity to enjoy the view.

Your destination is to learn programming and learn how to teach programming to your children. This book does exactly that. Starting from a well-crafted tabletop game called Namatreasure, this book introduces all the basic concepts of programming and all the types of instructions, using plain English which is readable and completely understandable by anyone who speaks English. These minor explanations are elaborated with examples that you can practice until you become familiar with them.

From the tabletop game Namatreasure, we continue smoothly to Scratch by MIT Media lab - a free programming language which is easy to use, enables sharing and is flexible and exciting. It is an environment that enables us to recreate the Namatreasure game and above all to create our own commands easily. By creating our own commands, we can make programming more like writing instructions to a friend, more like using the English language rather than the usual programming language. In Scratch you are free to investigate, and at the same time you are guided to reach your goal quickly.

We are not limited to Scratch though. We will also be investigating Blockly by Google which is a programming language that is used in many devices. Blockly is free and any company can incorporate it into its products. For this reason, it can be found in many educational robots. Blockly can also be found in more useful applications, like programming mobile phones. We will mostly use Blockly to test the level of knowledge and expertise attained. Blockly has created a set of demo games with 10 levels, each of which can be used to test someone's programming skills. This book will guide you to reach the highest level of the Blockly game, Maze, and be comfortable there.

The highest level of Blockly games is for advanced programmers and from there, one can proceed alone to learn other environments which might be more interesting and more suited to individual tastes and likes.

Because programming environments change continuously and because YouTube, Vimeo and similar applications enable us to share videos easily, we also use videos to provide the details in a more vivid way. Therefore, you can view the videos and follow the channel that is mentioned in the book, to see more related material. The book is your companion that you can take with you anywhere; the videos are the recipes that you follow to do the examples. You can also write to us and send us comments so that we can add more videos on things that interest you.

Moreover, in Scratch we have examples referenced in the book that you can copy (remix) into your account and work on them. There is also a Scratch account mentioned in the book, that you can follow to get access to more related examples.

Once you become familiar with programming, you can then proceed to our next book “Idea_Simulate_Develop_Explore_Build”, which is under development, and which will contain examples of smart devices that the whole family can build and program, as well as useful mobile applications.

Thank you for selecting this book, we are all very excited for you and will be here to guide you in many ways, as you go through the wonderful journey of learning and sharing programming.

More material regarding the book can also be found at:

www.patreon.com/namatreasure.

Notations

variable names are enclosed in ‘ ‘ for example ‘robotd’

Names of software buttons, on the screen, are enclosed in [ ] for example [Remix]

Software commands are enclosed in ‘ ‘ or () for example (**Go to cell X**[n] **Y**[n])

Key reserved words of a programming language are written in bold with the first letter in capital when that word starts a new command, for example **If, Repeat, For,**

[n] n in brackets stands for any integer

[direction] direction in brackets stands for any direction for example north, south, east, west

CHAPTER 1

1 Introduction to Programming as a New Language

- Programming is easy
- Programming is like any another spoken language
- Programming is easier than writing essays
- If you can communicate, you can write programs

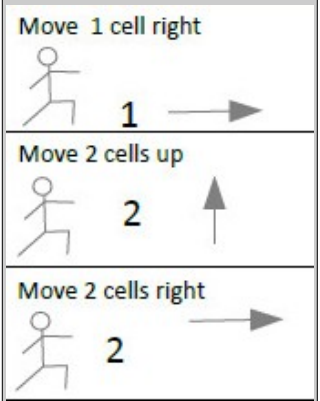
Before feeding their children, parents taste the food to see if it is too hot, cold or salty and then they give it to their children. You can proceed in the same way with this book. First, read and try the examples up to and including chapter four. Allow yourself to get comfortable with them. Do not worry, they are straightforward. Once you are comfortable in your understanding, then call your children to play and learn together. You can pretend that you do not know anything and that you are starting together, which is true because only with experience can you really learn programming.

1 Programming is easy

Programming a computer is much less complicated than most people think. Programming is like writing instructions with simple sentences. You can look at the example below:

Pic. 1 Task: The duck must reach the treasure box

Pic. 2 The program is in plain English and in symbols.



Programming is as easy as giving instructions in a sequence. Maybe you have learned that programming is done with 1's and 0's, or with complicated symbols and curly brackets. It is true that machine languages, which only the computer understands, are programmed with 1's and 0's and that third generation languages use curly brackets and many other symbols that might seem scary in the beginning. However, programming languages have evolved and now we are using the fifth generation and graphical languages. When we speak to each other, our brains understand words as electrical impulses that travel along neurons to be stored somewhere in the brain. Although this is a low-level task performed in the brain with electrical impulses, we operate at a much higher level without thinking about these low-level functions. Computers operate in much the same way and programming languages are becoming more like plain English. When we program a computer, we use words and simple-structure sentences. In fact, it is easier to write a computer program than it is to write a good essay!

A programming language is like any other language: it has a vocabulary, names that represent instructions that do something,

and a syntax (rules on how to put the instructions together). The syntax is very limited and there are only a limited number of ways that we can put the instructions together, so programming as a language is much smaller than spoken languages like English and Chinese.

2 What is a computer program and the use of repetition

A computer program is a set of simple instructions, much like giving directions to someone on how to go to a certain place or cook a certain recipe. One central idea in computer programming is repetition. The use of repetition does not bother computers at all, in fact, computers love repetition. By using repetition, we can understand things more precisely. When we want to do something several times, we can use the same structures every time. In this way, whenever we see these structures, we will immediately recognize what they mean. When you write an essay, you are taught to avoid repetition by using different words, phrases and structures, but computers don't mind repetition at all. I remember when I was studying at college; my roommate, who was a physics Master's student, would say he could not understand how I could write pages and pages of program. The truth of the matter was that it was a repetitive task and the original work was not very much at all.

3 Repetition and simple sentence structures

Programming uses very basic sentence structures and moreover, uses the same sentence structures over and over. Although the sentences are simple, there is one drawback – you cannot change the sentence structures any way you want. You must learn them as they are. Even if you write the structure just a little bit incorrectly, the program won't work. But there are syntax checkers which work much like an auto spell-checker in a word processor, which help you write the program correctly. Even more easy is that there are pre-made sentence structures that only require you to fill in the blanks in order to make the program. Simple sentence structures look like this:

Move 5 **steps** forward,	(example)
Move 4 **steps** up	(example)
Move NumberOf **steps** Direction	(sentence structure)
Turn right,	(example)
Turn left,	(example)
Turn Direction	(sentence structure)

Using these simple sentence structures, you can create your own sentence structures and new instructions, like:

Define ‘Move_horse’

Move 1 **step** forward

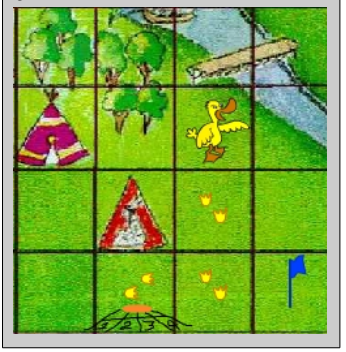
Turn left

Move 2 **steps** forward

End define

From now on you can use the instruction ‘**Move_horse**’, which is like the movement the horse does in Chess.

Pic. 2 The program is in plain English and in symbols.



> **MAGIC:** The magic of programming (yes, there is magic in programming) is that you can create your own sentence structures from the existing ones so that you can expand the language. Yes, you can become a linguistic professor and a new word creator. There is a lot of creating in programming and a lot of magic. Stay with us.

All these instructions are simple sentences. The most important thing in programming is to think logically and make a plan of what you want to do. Learning programming, helps in learning other skills like problem solving, breaking problems down into simpler problems, telling stories and effective communication.

4 Immediate feedback

When you create a program, you test it and know immediately whether it works. Testing provides immediate feedback. This is the beauty of programming. No wonder our children are so fascinated with tablets and computers; they get immediate feedback and when we say immediate, we mean in milliseconds, maybe less. The immediate feedback gives you the chance to fix your mistakes as soon as they are created, because mistakes always happen, and the important thing is to try again until you get it right. To fix the mistake, you must execute a program step by step like the computers do. Investigating what the computer should do when it executes a program is an excellent way to test your understanding of every instruction. Programs start to work correctly only when you know exactly what and why they do what they do. This is a striking

truth that I have realized because it happens to me all the time, whenever I write a program without knowing exactly how it will behave, it never behaves as I want it to. Whenever I study a program, analyze all its subprograms and know exactly what each instruction does, then it always works.

5 Logical plan

A program is like a plan. When you create a program, you are making a logical plan. It does not have to be fully developed or cover all the details because it can be changed as you write it. The important thing is to cut a complicated task into simpler sub tasks and then cut them again into even simpler tasks and so on, until you find a solution to each sub task. When you find a solution for each sub task, you begin assembling each one together until they are all included in one program that solves the complicated task. We will deal with complicated tasks and how to tackle them by decomposing them into simpler problems.

6 Anatomy of a program

Programming is separated into three basic phases: the initial conditions, the task to be completed (or the main loop) and the result (or the conditions of exit from the loop). Similarly, with cooking, first you gather the ingredients, (this is the initial conditions), then you start cooking and tasting along the way, (which is like the loop) and when it tastes right, you stop because you have reached the desired result (which is like exiting the loop).

6.1 Initial conditions

Initial conditions in a program, is like laying the ingredients on the table to be cooked, or setting up the pieces before a chess game, or filling the car with petrol before taking a journey. Suppose you are going to find treasure on an island. The initial conditions will tell you where the island is, and from which point you need to start. Unless you know the correct island and starting point, you will never find the treasure. Another example of initial conditions is the setting up of the pieces of a tabletop game, like chess. If the setting up is done wrong all else will be wrong.

6.2 The main program

6.2.1 Linear programming, as a story from start to end, describes what to do.

The next part of making a program is setting a list of tasks and repeating them until the result of the program is obtained. In our example on finding Namatreasure, the program is a list of instructions to follow. These instructions give directions, information warnings and options to select. The main program can be a list of instructions that are executed one after the other, like a list of things to do. An example of instructions that go from the first to the last, are the instructions for searching to find a treasure given above, which are like a story or a catalog of instructions which take you from one point to the next.

6.2.2 Loop programming, repeating continuously, describes how to react.

The main program can also be a repeat loop that contains instructions which are continuously executed at each pass when their turn comes. This kind of program has a main loop where the instructions are repeated continuously and instead of instructing what to do, they instruct how to behave. For example, you can program a computerized character such as a duck to swim when it enters water and walk when it leaves water. This is achieved with an "**If**…….**then**……..**else**….." statement. Similarly, in cooking, you cook and taste and repeat, until the food is cooked. Cooking and tasting are in a loop and tasting is inside the '**If**' statement: '**If** (it tastes cooked) **then** stop cooking **else** continue cooking.'

Another example of a loop is playing chess with the computer. The computer is on a loop and at each turn it decides its next move until the game finishes.

6.3 The result, the outcome

Continuing with our treasure example, the last instruction is (**Dig until** you find the treasure box). If you dig and do not find the treasure, then

you know that you have done something wrong and you must search for the mistake, otherwise you are finished and you have found the treasure.

For the programs that use the loop and prescribe behavior, there is still an end and a condition where the program ends, for example when the character reaches its goal.

7 Programming Environments

What is a programming environment? A programming environment is like a tabletop board game. The programming environment is the board, some pieces which move on the board and a set of rules. From our experience, we understand different real environments immediately and prepare to handle them accordingly. In programming, the same applies, although the environments are not understood so perceptually. In this book three programming environments will be introduced:

Namatreasure, Blockly and Scratch, so that you can appreciate that programming is not tied to only one environment. Generally, there are programming environments to program mobile applications (Blockly - MIT App Inventor), to program robots (Engino ERP), to program games on the internet (Scratch, AgentSheets), to program home automation (Arduino) and so on. All programming environments share a lot of common elements and programmers can easily move from one to any another easily, since they have the foundation.

Programming is very easy to do when there are videos with step-by-step instructions. We provide these step-by-step videos to keep the book simple and short. The good thing about the videos, is that you can repeat them any time you want. If you spot a missing link that was not explained fully, you can always write to us.

> **KEY CONCEPTS:** Programming is easy, there are a limited amount of structures and they are repeated all the time. Computers love repetition. Mainly programs have two parts. In the first part everything

is setup and in the second there are instructions to be executed once or many times.

CHAPTER 2

2 Namatreasure Programming Environment

- Get into the adventure, have fun
- Play with friends
- No computer needed, you play the role of the computer
- The easiest programming environment

1. Have fun, enjoy and get interested

It is critical to have fun, enjoy and get interested while you learn programming. It is fun. You can enjoy the things you create, especially when they come ‘alive’, and you should get interested because this is the most important skills your children will acquire. If what the children are doing is not fun and does not make them happy, they won’t do it. The most important thing is to attract the attention of the children and then keep it! To do this, you must make it fun, but first of all, YOU yourself should have fun. Equally important is that you should be excited because you will get to play and bond with your children.

We start with a tabletop game which explores the popular and exciting theme of a treasure hunt. At the same time, we are also looking at two other programming environments: Blockly Games and Scratch, which are graphical, fun and easy to start. The other two environments, Scratch and Blockly Games, are free thanks to by MIT and Google, as well as the USA government that funded related research.

You have to set an environment that is pleasant, and you have to be part of the environment because your children want to spend time with you. You have to be there. You have to offer yourself to the game so that the children will want to play. The environment is the playground where the treasure hunt adventures are going to take

place. This is a game. It is not teaching a class and it is not about programming or learning. It is about play and most of all, it is about bonding. No matter what you think, or what you want, you need to see it from the child's perspective.

2. Invite friends, create an interesting environment

In addition to giving yourself to the game, there must be an interesting environment that can be enriched by also inviting your children's friends.

An interesting environment is one that sparks the imagination, which helps children visualize things, but is also familiar so that it reminds them of things they already know. This association is necessary so that new knowledge can be connected to rooted knowledge. Rooted knowledge is easily accessible, whereas new knowledge may create an island of new knowledge in the mind that can be easily forgotten.

My favorite story for creating an interesting environment is the 'treasure hunt'. The treasure hunt happens on an exotic island and the environment has to have attractive graphics. 'Seeking the Namatreasure' has been created to offer a tabletop game with attractive graphics that look like children's drawings, so that children feel comfortable with them.

Namatreasure tabletop game will be used as the first step in playing with your children. Always keep putting yourself in the position of the child and imagine what they really want so that you can give it to them. Children first and foremost want to play games and have fun. Learning programming is something that we, their parents, want, therefore we should keep it in the back of our minds.

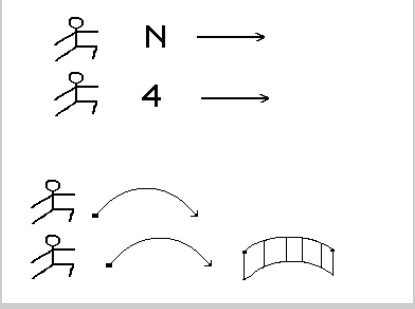
The importance of graphics cannot be stressed enough. They are hugely important and evidence of this can be seen with the large amount of money spent on advertisements, which are mostly pictures and movies. Also, it can be seen by the status given to graphic artists of children books which is equal to the status of the writer.

3. The simplest, easiest programming environment, Namatreasure

This first environment does not require a personal computer, it is the simplest environment and builds more on interpersonal interaction, than on interaction with the computer. It builds more on the foundation than on immediate skills and it also teaches children the most basic skill in programming. The skill of paying attention to detail and precision, of executing every step of the program correctly, the way the computer does. Programs are composed of many steps that are easy to follow, but every step must be executed correctly in order to succeed. It also teaches the skill of looking at something from the perspective of the other person, which, in turn, is very helpful in the development of interpersonal relations. You will see that when you learn to interact with computers, you also become better at communicating with people. For example, the first thing a master computer does in a network of devices, is to communicate with each device and verify that they are all healthy and ready. How many managers do you know that do the same for their personnel?

Pic. 4 Working Environment Namatreasure

Pic. 5 Types of Namatreasure graphical instructions



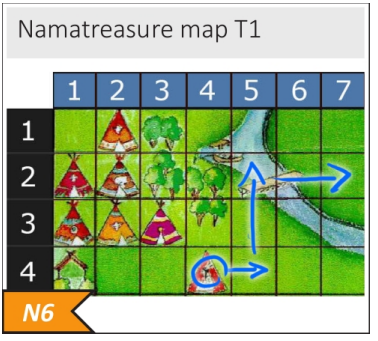
This game is a bridge from spoken English to programming English.

The map shown above is the working environment, where the treasure hunter travels to find the treasure. In illustration N5 there are some instructions that are used to guide the treasure hunter. In its simplest form, the game is played by two players, where one is the treasure hunter and the other is the treasure owner. The treasure owner hides the treasure in one of the squares, and then writes instructions that can lead the hunter to the treasure. The instructions are like those used in a programming language and are similar to those used in Scratch and Blockly. The treasure hunter receives the map of the island and the instructions for the treasure and follows the instructions to reach the treasure. The treasure hunter replaces the computer and acts as though he is executing the program. The role of the treasure hunter is extremely important because it helps children understand how the computer executes instructions.

If the children are between the ages of 6 and 9, you can dramatize the adventures and imagine that you are actually in a boat or walking in the forest and see the Indian huts. Drama for these ages works very well.

Examples of Namatreasure Hunts are listed below, starting from the simplest, to the more complicated. The level next to each example denotes the complexity level of the treasure hunt. There will be more elaboration on levels in the next chapters.

4. Treasures



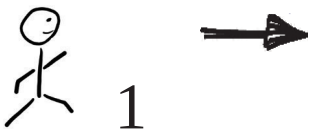
Treasure T1 (Level 1)

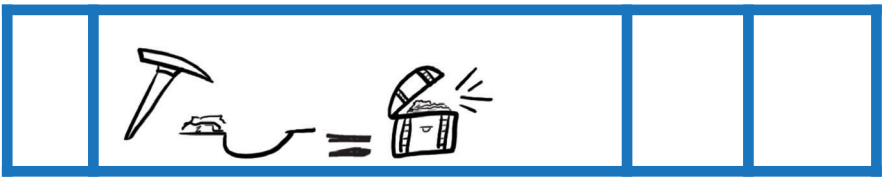
Mark the start with a rectangle or a circle, each blue line is one instruction.

(Watch video T1 on YouTube **https://youtu.be/FUmiJW_GkRg**)

The instructions are written on paper, on a template. All cells that have a white background are filled in by the user. The instructions can

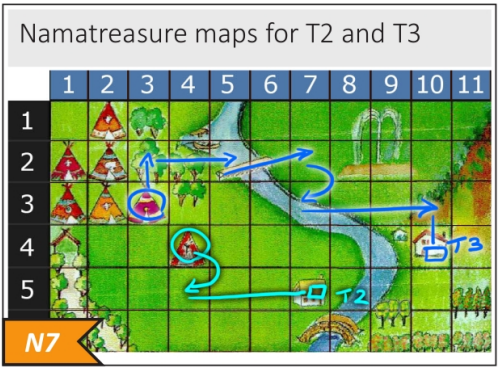
be written in words if the children can write words and in symbols if they cannot. If they are written in symbols they can be read by children who do not speak English. Symbols are a universal language.

Instructions that will lead you to the treasure (Program)			
Treasure Name	T1		
Treasure Island	Island of two rivers		
Initial position in X (column)	4		
Initial position in Y (row)	4		
Num.	Instruction	X	Y
1	**Move** 1 **cell** right 	5	4
2	**Move** 2 **cells** up	5	2
3	**Move** 2 **cells** right	7	2
4	**Dig until** you find a treasure box	7	2



Treasure T2 (Level 2)

Treasure T2 is at a higher level than T1 because it uses instructions without numeric values. It uses the instruction type, **Move** (direction) **until you** (condition). This simple instruction type is very common in English, but in programming it is one of the more advanced structures. Instruction type is the general form of the instruction where the words in parenthesis will be replaced when we write specific instructions. The word direction can be replaced with all words that show the basic direction for example right, left, up, down, or using more advanced directions such as north, south east, west and the diagonals of these. The condition can be replaced with a task to be completed, for example - reach a house-, -reach the river-, -see bridge on your right-, and so on. You can also think of the condition as a question. For example, “Have you reached a house?”,

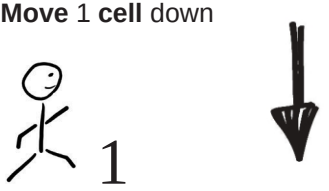
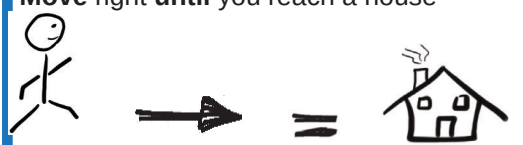


“Have you reached the river”, and so on. When the answer to the question is yes, then the condition becomes true. When the condition becomes true, the instruction is finished. Once an instruction is completed, we move to next instruction. The words in bold define the structure type and they cannot be replaced with similar words, nor can their order be changed.

T2 starts from an Indian hut at cell X (column) 4 and Y (row) 4, and from now on we write cell (4, 4) for ease of use. The first part of the movement is to move one step down. The second is to move right until you find a house and third to dig until you find the treasure T2.

Let's write the treasure in a graphical language that anyone can understand so that you can share it with all the children in the world.

Watch video T2 on YouTube **https://youtu.be/gOhTYHmQdGg**

Instructions that will lead you to the treasure (Program)			
Treasure Name	T2		
Treasure Island	Island of two rivers		
Initial position in X (column)	4		
Initial position in Y (row)	4		
Num.	Instruction	X	Y
1	**Move** 1 **cell** down 	4	5
2	**Move** right **until** you reach a house 	7	5
3	**Dig until** you find a treasure box	7	5

Treasure T3 (Level 3)

Treasure T3 is a higher level than T2 because it has more instructions. The instructions are the same as in Level T2 but because it is longer, it provides an added difficulty for children. It is difficult for children to execute one instruction and then forget it and concentrate entirely on the next one. They keep thinking about all the instructions and get confused. Therefore, they should practice a lot with longer programs. With practice they will learn to handle a complicated task piece by piece.

First, we will write the instructions for each path segment of treasure in T3 in short form because by now you are already a little experienced.

Instructions for treasure T3

Initial location before we start following instructions is Cell (3,3).

Watch video T3 on YouTube https://youtu.be/gm6H1uXN71k

The words in bold do not change in a type of instruction. The other words can be taken from a list of optional words.

Move 1 **cell** up.

Move right **until** you reach a bridge

Move across the bridge

Move 1 **cell** down

Move right **until** you find a tree

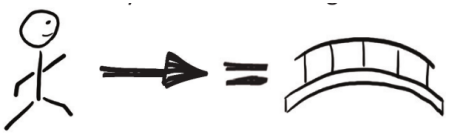
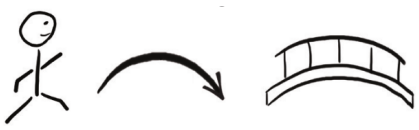
Move down **until** you reach a house

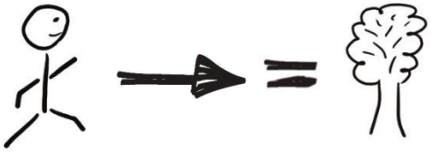
Dig until you find the treasure box

Note instruction (**move** across the bridge) means that the treasure hunter will move up to a cell where there is no bridge cell (7,2). Instructions that will lead you to the treasure (Program).

Instructions that will lead you to the treasure (Program)

Treasure Name	T3
Treasure Island	Island of two rivers
Initial position in X (column)	3
Initial position in Y (row)	3

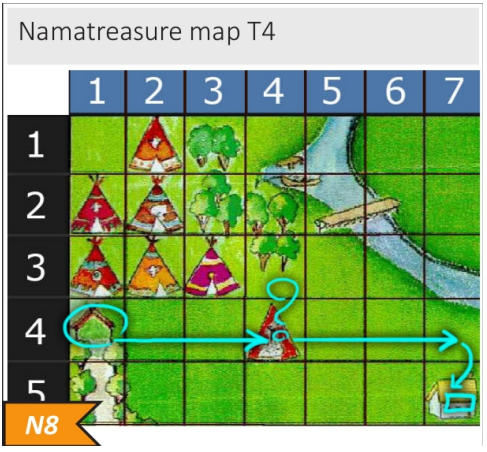
Num.	Instruction	X	Y
1	**Move** 1 **cell** up	3	2
2	**Move until** you reach the bridge 	5	2
3	**Move** across the bridge 	7	2
4	**Move** 1 **cell** down	7	3
5	**Move** right **until** you reach a tree		

		10	3
6	**Move** down **until** you reach a house	10	4
7	**Dig until** you find a treasure box	10	4

Treasure T4 (Level 4)

Treasure T4 introduces a new type of instruction called **If**. Follow the instructions for the treasure T4 and see if you encounter any difficulties.

(Watch video T4 on YouTube



https://youtu.be/Q1MFmynqK80

Treasure Island of two rivers

The instructions are not difficult to follow, but they have two new types of instruction. One is the '**Information:** ……'

that tells us some information to make the treasure hunt more interesting; to make it like a story; to add to the unexpected and

Instructions that will lead you to the treasure (Program)			
Treasure Name	T4		
Treasure Island	Island of two rivers		
Initial position in X (column)	1		
Initial position in Y (row)	4		
Num.	Instruction	X	Y
1	**Information:** You have enough food to go find the treasure and return	1	4
2	**Move** right **until** you find an Indian hut	4	4
3	**Information:** In the Indian hut you meet the Indian chief, who likes your food very much and asks you to give him some to taste	4	4
4	**If** you give the Indian chief food t**hen** the Indian chief will give you a key **else** you will have enough food to go and come back	4	4
5	**Move** right **until** you reach a house with yellow roof just below you	7	4
6	**Move** down **until** you reach the door of the house	7	5

7	**If** you have a key with you t**hen** you can enter the house find the treasure find the food **else** you go back without the treasure	7	5
8	**End of treasure** T4	7	4

to enable us to use the '**If'** instruction. The second new instruction type is the

"**If** ..... **then** .....**else**……**.** ".

It can also be written as

If (condition)

then do something that is written here

else do something different that is written here

It is much easier to follow the instruction than to describe what type of instruction it is. The '**If'**.. is not a type of instruction, it is a whole structure because it contains instructions and, based on the condition, it guides the flow of the execution of the instructions to one of two paths. The '**If'** structure is like a T-intersection on the road. At the T-intersection you completely stop and then you decide which road to take; you cannot take both.

The '**If'** structure is very important in programming. There is even one programming environment called AgentSheets, where all the instructions are of the If type.

Treasure T5 (Level 5)

Treasure T5 introduces one more element that we will see after you follow the instructions.

Watch video T5 on YouTube https://youtu.be/ggZgUv9zpp8

It is easier to execute the instructions of Treasure hunt T5 than it is to explain it theoretically.

Main Block of instructions that will lead you to the treasure (Program)			
Treasure Name	T5		
Treasure Island	Island of two rivers		
Initial position in X (column)	4		
Initial position in Y (row)	10		
Num.	Instruction	X	Y
1	**Move** 4 **cells** up	4	6
2	**Move** right **until** you touch the bridge	7	6
3	**Move** across the bridge (Move until you no longer touching the bridge)	9	6
4	**Move** right **until** you reach a river	12	6
5	**If** it is summer t**hen** **follow** the Block “Cyan” instructions	12	6
6	**If** it is winter t**hen** **follow** the Block “Grey” instructions		
7	**Dig until** you find the treasure		
8	**End of treasure** T5		

Sub Block of instructions of T5 Cyan			
Num.	Instruction	X	Y
1	**Move** up **until** you reach a house with a yellow roof	12	3
2	**Move** 3 **cells** right	15	3
3	**Move** 1 **cell** down	15	4
4	**Return to the previous block of instructions**		

Sub Block of instructions of T5 Grey			
Num.	Instruction	X	Y
1	**Move** 3 **cells** right	15	6
2	**Move** 2 **cells** up	15	4
3	**Return to the previous block of instructions**		

Treasure T5 starts from the Main Block of instructions, continues to other blocks of instructions and then when the execution of their instructions finishes, the user returns to the instructions of the previous block. It is very easy to perform these tasks in Namatreasure because they are things that we do all the time. However, In the official programming languages, these simple structures may confuse some children, so it is good to build strong foundations in Namatreasure. In the programming environments we will investigate in this book, the transition from Namatreasure will be very smooth because they are very advanced environments and because we will redo the Namatreasure Treasures T1 to T4 in the Scratch environment.

Namatreasure is derived from the word Nama and the word treasure. The word Nama in Greek means the source. Namatreasure stands for the treasure that your children will find at the source. The source is the programming language that will enable us to talk to the intelligent devices that surround us, but above all, to understand when someone is trying to program us.

MAGIC: By practicing programming you acquire the skill of attention to detail and precision, executing every step of the program correctly, as the computer does. Programs are composed of many steps that are easy to follow, but every step must be executed correctly in order to find out how it will function. It also teaches the skill of looking at something from the perspective of the other person, which can also be very helpful with the development of interpersonal relations. You see, when you learn to interact with computers, you also become better at communicating with people.

KEY CONCEPTS: Programming is fun especially if your friends join you. There is an easy way to start even without a computer, by playing the educational game Namatreasure. You can start programming by writing instructions to find hidden treasure. The programs are executed by your friends when they try to find the treasure you have hidden. All instruction types that are used in most programming languages are similar and they are also used in the game.

CHAPTER 3

3 Computer Programming Environments - Scratch and Blockly

- From paper to screen
- Programming environments are like board games
- Each programming environment is targeted to a specific audience and a specific task
- Once you have mastered one programming environment, you can learn the others
- Exploring the programming environments is fun

1 Let's make computer programs

Congratulations!! If you are reading this, it means you have come a long way. If you have practiced writing treasures and finding treasures, then you already know programming. You have all the basic concepts down, now all you lack are the specifics of the programming environment.

Therefore, in this chapter, we will move from paper to screen, to real computers and mainly to the network. We are moving to the network because what is the use of having or knowing something, if you cannot share it.

Once we have managed to play with our children, make them interested, find treasures and compete, what's next? A tabletop game and paper are the foundations. But up until now, they have only worked with their imagination. There were no errors, and everything worked, but it wasn't exactly programming because

it did not involve a computer. It was the foundation of programming, and a foundation without the building is nothing. Now we must proceed smoothly to the building.

Fortunately for us, there has been an enormous amount of effort to make programming easy for children. The US government (public sector), private enterprises and academia have put a lot of resources into creating tools that make programming challenging, stimulating and attractive to children. This is because they consider programming to be one of the basic foundations for science and for the development of a prosperous nation. The important thing for us is that there are basic tools to help us learn programming easily. We will use those tools as an extension of what we have already learned, and we will use what we have learned to transpose it into programming on the screen. It won't be exactly the same, but it will be very close. The principle is that we want to transition smoothly into a higher level of learning.

Each programming language (and there are many) does something well and effectively. We will use the programming languages that are most effective for us and we will explore the most innovative and interesting things we can achieve with them. Do not expect to easily transpose one problem to all programming languages. We will find this out soon.

In the next chapters we will see the Namatreasure game in a programming environment, where we can easily play any treasure hunt and even upload it to the internet so that everyone can play. The two different programming environments that we will investigate are free, graphics oriented and intuitive. Each one has different advantages, and understanding the strengths of each programming language is the best way to solve a problem.

Like spoken (natural) languages, there are many programming languages. We are constantly inventing new ones in the hopes that with the least effort, there will be a transfer of knowledge and know-how from human beings to computers. We won't learn all the programming languages; there is no need. We have learned the foundation and from there it is easy for us to learn different programming languages.

In real life, we don't know which programming languages we will use, or what programming languages will still be used. Learning a new

language is just a matter of learning a few basic data structures, syntax rules and some vocabulary. The important thing is the skills, and this is what we will master and accumulate. Now we are ready to test what we have cultivated.

First let me tell you about my personal experience with programming. After I graduated from my Masters program, I counted 15 programming environments that I had used during my studies and in my work up to that time. There were general programming environments like Pascal and C; programming for electronic devices using assembly language and PDP 11; programming in mathematical environments; programming in statistics; programming for factory simulation and network simulation; for robot programming; for CNC programming and for industrial electronic controllers. Now, I have to learn new programming languages and environments, but I know programming, so I only need to learn the new specifics. It is like driving a car, every time you enter into a new one, you have to learn where everything is, and it takes a while before you feel comfortable.

2 Programming Environment Blockly Games

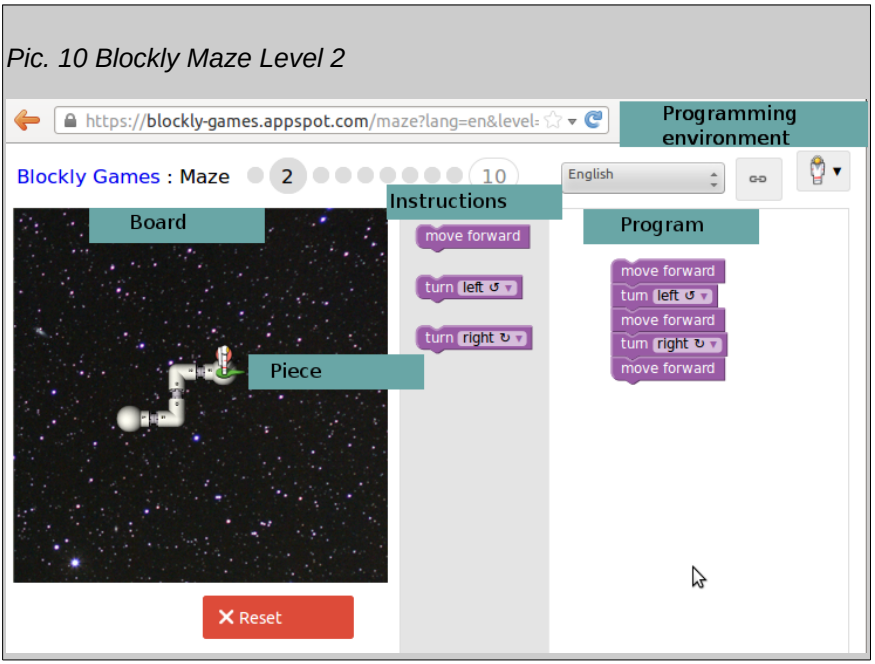
Let's proceed to write our first program in the programming environment of Blockly Games. Blockly Maze Level 2.

Portions of this page and the following pages referring to Blockly are reproduced from work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License.

The original pages can be found at:
https://blockly-games.appspot.com

The whole programming environment is shown in the picture below.

Pic. 10 Blockly Maze Level 2



In the right side of the picture above, the five stacked sentences are the program in the program area. In the middle part, there are the instructions the programmer can use by dragging and dropping them

into the program area. The instructions are needed to solve this task. The left part of the screen is the board (play area) in which the program acts and where the pieces move.

In the initial conditions, the astronaut stands in the left lower sphere, as shown in the picture.

The main program is the five instructions stacked one below the other. The instructions are executed top to bottom and the astronaut obeys the instructions and moves.

The aim is reached when all the instructions are executed, and the astronaut has moved up to his destination – the red pin flag in the upper right sphere.

Please note that this is a complete program and a task has been achieved. This is a Basic Level program, called Blockly Maze Level 2.

We will use levels throughout the book so that you can measure your progress, record your achievements and see where we are heading. The first instruction, '**move forward'**, is the simplest type of instruction. The second instruction, '**turn** left**'**, is also very simple and the fourth instruction, '**turn** right**'** is the same type as the second. The first instruction is used three times. Please note the '**turn** left' and '**turn** right' instructions have a downwards facing arrow at the farthest right part of the screen. When this is pressed, a drop-down list appears containing the directions left and right. Similarly, other instructions will have other choice lists. By using choice lists, the number of available instructions in the middle sections is less and can fit.

Blockly Maze is one of the easiest programming environments. (Watch video BM2 Blockly Maze Level 2 on YouTube https://youtu.be/eFPfMshwvyo)

Link to Blockly Games https://blockly-games.appspot.com/maze .

3 Programming Environment Scratch

We now proceed to another similar graphic environment, but this one is made especially for children.

“Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See http://scratch.mit.edu”. “Scratch is a programming language and online community where you can create your own interactive stories, games, and animations - and share your creations with others around the world. In the process of designing and programming Scratch projects, young people learn to think creatively, reason systematically and work collaboratively. It is available for free at http://scratch.mit.edu”.

This programming environment has been influenced by Logo, Smalltalk, HyperCard, StarLogo, AgentSheets and Etoys programming environments. As you can see it is based on many languages and has gathered the best of each. In my opinion it has a friendly easy-to- use interface, it is free, it can run directly from the internet and it has a strong support group. Above all, it makes it easy to share and collaborate.

It is aimed at attracting children to programming and in order to achieve this goal, the creators had to provide an environment where, with little effort, they could create an exciting graphical game. Coding the program should be easy and the program itself should be readable and understandable, meaning that users should be able to read it as though they were reading a book and understand what it instructs the program to do. Each game can then be shared over the Internet with friends and followers. By the end of this book, you will learn how to teach your children to make arcade games like “Super Mario” and to program robots that compete against other robots.

This environment is very similar to Blockly Games. In the right part of the window is the program.

In the left part are the instructions the programmer can use. The instructions are separated into ten categories shown at the extreme left part. In this screen shot, the category Motion is activated, and the instructions shown are from that category. The instructions in the program are the same as those used in Blockly and the two programs do similar things, although not exactly the same.

https://scratch.mit.edu/projects/93614537/editor

[pic N11 Scratch Programming Environment T1]

Pic. 11 Scratch Programming Environment T1

The instruction ‘turn right’ in Blockly Games is the same as ‘Turn right’ in Scratch and the instruction “move forward” the same as “Move 1 cell”.

The result of the program is shown in the picture. The cells where the duck will walk are shown with cyan circles. In this program there was also an initial condition, the program and the result. You can remix this program and start from a different cell and end at the target cell where the gift box is.

Instructions fit like a puzzle so the user cannot put them in the wrong place. In the left part of the window is the board of the working environment. The board is a graphical window where the pieces (they are called sprites in Scratch) can move and behave according to their

program. The user can design or import sprites. These sprites can be animated by the program. Similar programming environments call the pieces actors, agents or characters. You can call them whatever you like, the important thing is that you can program them to do a specific task or you can program their behavior and let them interact with each other and the user.

It is obvious that this environment is more complicated and little children need guidance to enter it with confidence. It is like small children holding their parent's hand when entering a fun-park. As long as they are holding their parent's hand, they feel safe and enjoy it, but if they lose sight of their parents they start crying.

Follow the link below to go to that project.

Watch video ST1 Scratch treasure 1 on YouTube

https://youtu.be/Ub3Rc7J1fmM

The project described above was specifically made for the book to link Namatreasure with the Scratch environment. The instructions shown in purple, are made by the user using simpler instructions in Scratch. We made the purple instructions by writing a simple program using the original instructions of Scratch, so, we have extended the vocabulary of Scratch.

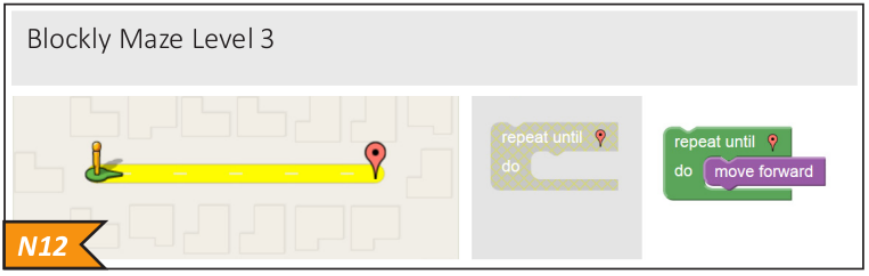
Go to these locations to practice with Scratch and Treasure T1

https://scratch.mit.edu/projects/93614537/#editor

4 When the main program is a loop

After having seen some examples of programs and gained some experience, it is time to see the second type of main program that we talked about in chapter 1.6.2. Please note this type is the most widely used. It is a Loop where instructions are repeated forever, or until something happens.

In the following example, the traveler has a target to reach – the red flag (pin). The program to guide the traveler to the target is composed of only two instructions, but one instruction is different, it is not like the normal instructions which are piled one after the other. In plain English, this instruction could be '**Move** forward **until** you reach the red flag'. In programming language there is a small twist, so that it becomes '**Repeat until** you reach the red flag'. It contains all the instructions that are to be repeated.



In this case there is only one instruction 'move forward'. In the illustration N12, in the graphical Blockly language the '**repeat until** you reach the red flag' embraces all the instructions that are going to be repeated. The instructions are executed as long as the target is not reached. The way you think about this instruction and the way the computer executes it is:

First check if the target red flag is reached. The answer is "No" (false), therefore the instructions inside the embrace of '**repeat until** you reach the red flag' are executed.

The traveler moves one step forward and following the green line of the command '**repeat until** you reach the red flag' he goes back to check again if the target red flag, is reached. The answer is "No" again, so the instructions inside the embrace are executed once

again. We continue to go around the loop until the traveler reaches the red flag. Then the answer to the check is “Yes” (true), so that the instruction within the embrace is not executed and execution moves to the instruction after the ‘**repeat until** you reach the red flag’. In this case there are no further instructions and the program is finished.

The main difference between

Move forward **until** you reach the red flag
and
Repeat until you reach the red flag
    Move forward
Return to repeat

is that inside the ‘**Repeat until**…….**Return to repeat**” we can have more than one instruction. Another important difference is that the first type ‘**Move** forward **until**..” is more readable and understandable by people because it is used more widely.

Watch video BM3 Blockly Maze Level 3 on

https://youtu.be/S8lX-M0x11Y

Another example of a program with a LOOP is the following:

Repeat forever
    Lights ON
    wait 1 **second**
    Lights OFF
    wait 1 **second**
Return to repeat

In the above code, there is a Loop that repeats forever. The words ‘**Repeat forever**’ stands for ‘**repeat forever the instructions below**’. The instructions above have the effect of turning on and off a light every one second.

The program in illustration N13 is from Scratch and is for the sprite flag of the previous example. It makes the flag change costume. In

this graphical representation of the Loop instruction, it is shown very clearly which instructions are in the loop. However, the word '**forever**' does not convey a clear message, and therefore you can think of this instruction as the same as the '**Repeat forever**' command.

LINK https://scratch.mit.edu/projects/101656127/#editor

In the project see the code for sprite Flag.

Pic. 13 Flag Code, Loop in structure in T1

Example of an exit from a Loop using comparison

Count = 0

Repeat until (CountTimes > 1000)

    Lights on

    Delay 1 second

    Lights off

    Delay 1 second

    I**ncrement** CountTimes **by** 1

Return to repeat

Next Instruction

Another example of a "**Repeat until**….." is shown above where the condition is a mathematical comparison. The condition is in parentheses. In this example the condition is true (that is, it is correct) for all the values of CountTimes >1000. When the criteria are false, the four instructions within **Repeat until** ….and **Return to repeat** are executed repeatedly. When the criterion becomes true the instruction after **Return to repeat** is executed and the program execution proceeds.

Watch video ST2 Scratch – Namatreasure Level 2 on YouTube

https://youtu.be/TR5m9j4HNAo

Take a deep breath, if you have reached this far you now have a basic understanding of programming, but as they say the beginning is 50% of the work to be done. Relax; from now on you will just perfect the art and science of programming.

Watch video BM3 Blockly Maze Level 3 on

https://youtu.be/S8lX-M0x11Y

Watch video BM4 Blockly Maze Level 4 Find the repeating Pattern on YouTube https://youtu.be/YLdbX4oFpuw

Watch video BM5 Blockly Maze Level 5 separate motion in phases, linear phase loop phase on YouTube https://youtu.be/Xxg8cyLIQMs

MAGIC: In the latest programming environments, Blockly Games, Scratch and others, instructions fit like a puzzle so the user cannot put them in the wrong place. The user does not have to write even one line of code. Programming is done by dragging and dropping instructions from the tools section to the program area.

KEY CONCEPTS: Programming technology is progressing fast. The new programming environments have ‘drag and drop’ facilities, to write the programs. Any syntactic errors are immediately spotted so that the user can correct them. With little effort, the user can create thrilling graphical games. Imagine the future of programming environments. Programming could be achieved by talking to the computer as though you are two good friends sharing a cup of java coffee.

CHAPTER 4

4 Tips to make an environment interesting

- Measure progress with badges, levels and progress bars
- Reward every bit of effort and let learners explore
- Get feedback, explore, find the problem, (debugging)
- Insert uncertainty, an element of luck
- Share with others, work as a team
- Play to learn programming

Introduce computer game rules to make a game interesting. Remember that teaching children programming in an interesting way, can only be done using games.

We can use the accumulated knowledge of what makes computer games so attractive, to help us make our game attractive too. Computer games have a theory and certain rules which make them very interesting. There are some basic rules present in all games. They have been developed through the creation of many games over time, and most games follow them. You cannot escape these rules. Instead you need to follow them, not only because they make the game fun, but also because they work. Let's see how we can use these rules to accomplish our purpose.

1 Levels, bar to measure progress, badges

Everything you do must be done in small easy steps, as if there is a ladder that can be clearly marked so that children can see themselves climbing, progressing and achieving something. The Blockly Maze game is an excellent test for measuring progress at the basic

programming levels. The complexity of the programs done so far is equivalent to Level 5 in the Blockly Maze Level System.

Learning is like a mountain that we can separate into several small trips, where each trip is divided into levels. Ten levels is a good number; not too large and not too small. With each trip we climb higher up the mountain and explore new areas. For example, Blockly Games has several trips which are called games. The basic game is the Maze, then there is the Bird, the Turtle, the Movie and the Pond.

You have to be very careful and make sure that each step is mastered before moving on to the next. Each step starts with a repetition of what has been demonstrated and learned in the previous level. Each piece of new material is covered several times and at each level there is an activity that helps you practice all the material covered. However, you can still proceed to the next step to make it more interesting, but steps must be repeated in order to be mastered. We will use Blockly games and levels to show the knowledge and experience that your children will acquire, because they are universal, they are excellent and can be used for comparison across the world. We will also use the Treasures of the Namatreasure game because that is the easiest level that is possible. Remember this book is not about teaching or learning but about playing – learning comes automatically without effort. Each step is a small increment. You need to carry the children along with you. You cannot lose them. You cannot make big steps. Repeat the steps as many times as needed in order to master that knowledge through practice. The good thing about the videos is that you can repeat them any time you want. If you still spot a missing link that was not explained fully you can always write to us.

There are two accomplishment bars when it comes to programming. One is the level of understanding of written programs, and the other is the level of writing programs from scratch. These accomplishment bars of understanding and writing are very distinct in programming and each child will be at a different level in each bar. Usually most people who learn programming proceed much faster in the understanding bar than in the writing bar. The understanding bar focuses on the levels of complexity of the instructions that children understand, as well as the instructions they can change in order to make the program a little different. The writing bar focuses on the level of complexity of the instructions that children use in their own programs that they start from scratch (nothing). In Namatreasure the

two accomplishment bars are very clear. In the first accomplishment bar, children are treasure hunters and follow the instructions that the treasure owner has written in order to find the treasure. In this role children can quickly move up through the levels. In the second bar, they are the treasure owners and they must write the instructions for others to follow. In this role, and depending on the age, it might take a year to move up a level, and this has to do with their ability to deal with abstract concepts. It is the same with mathematics where they are taught concepts according to their age.

https://scratch.mit.edu/projects/93609616/ project T2 in Scratch, explore it

We have shown Blockly Maze Level 5

Watch video ST2 Scratch – Namatreasure Level 2 on YouTube

https://youtu.be/TR5m9j4HNAo

2 Reward every bit of effort and accomplishment and don't punish failure

Think of failure as a hypothesis that proved to be wrong; as acquiring knowledge of the things that do not work. Let us make a quick note on this. First year university students are taught an introduction to programming, and many find it very difficult. Recently, with the help of graphical programming environments made especially for children and with the help of this book, we can now teach programming to six-year-old children. So, have a little patience; it can be done, and it can be fun. Try not to get alarmed about making progress, just stay cool.

This rule is entirely up to you and the attitude you have towards the effort of your child. Environments like Scratch, Blockly Games and Namatreasure are not exactly video games where you are being led all the time. The programming environments are more open, and you have complete freedom to do whatever you like. Therefore, it is up to you – the leader that leads the child on this wonderful journey – to keep this rule in mind. I must give a warning here. Most people do not have this attitude of rewarding effort and ignoring failure. Unfortunately, some parents expect their children to know something before they are even taught it and some parents expect their children

to know something after only having been exposed to it once. The truth is that learning comes with repetition and practice and that even if you are a master of something you can still make mistakes.

I still remember a conversation that happened 25 years ago between my professor and a fellow student.

The professor said: “You do not seem to fully understand this paper,” to which the student responded: “But I have read it two times.”

Professor: “Two times? You have to keep reading it until you understand it even if it takes five or ten times: you read it until you understand it.”

This professor did not only teach this concept, but he also actually practiced it. There was another situation where he was talking to one of his PhD students and I was waiting my turn. The PhD student was trying to explain a process of selecting the best among many and the professor did not understand, so he kept asking the student two main questions. The student was trying to explain using different words each time, but the professor kept asking the same questions. There were more than five rounds of this questioning and answering but the professor did not for a second pretend that he understood. This kept going for more than 30 minutes, without the professor reaching any understanding at all. What most people do is pretend that they understand what you are telling them, so there is no real communication. I share this example in order to stress the fact that doing something once or saying: “I understand”, does not always mean a lot. Usually books on programming condense the essence of programming into one or two pages at the front for reference. These two pages don’t include all the practice it takes in order to master the art. Practice and repetition do the trick, but the most difficult part is to attract someone to this repetition, which is what this book is all about. It is about motivation, techniques and spending time with your children, practicing programming through games and play.

3 Feedback, Explore, Find the problem, Debugging

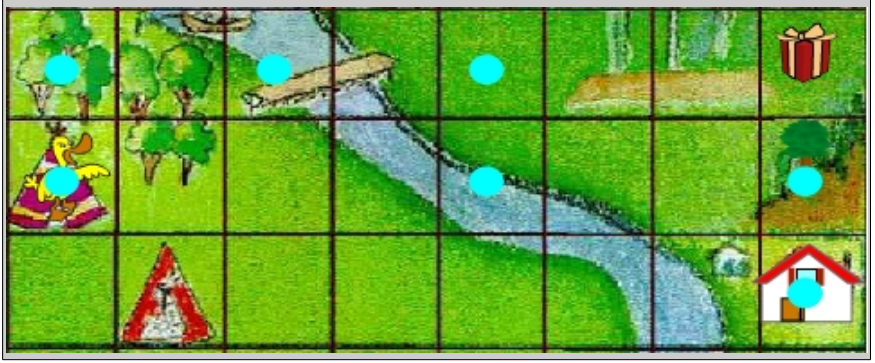
In school and at work, when you take a test or submit an assignment, the feedback usually comes when you have already forgotten what you have done. Programming has immediate feedback. After you finish writing the code and run the program, you get the result. In

Namatreasure, once you have followed the instructions and reached the last one, which says: ‘**Dig** to find the treasure’, you get feedback. You know if you have found the treasure or not. In Scratch programming, if something goes wrong you will know about it as soon as you run the program because it will not perform the way you expected it to. Not achieving the goal from the first attempt is common in programming, which is why there is a famous computer term known as debugging, meaning to find the faults and mistakes. The term debugging, in the early computers that used to run on relays, meant to remove bugs from under the contacts of relays. The bugs under the contacts prevented the current from flowing, causing the program to malfunction.

Today, debugging means to find the mistakes in the program that are causing it not to perform as planned. Debugging is one of the basic skills learned in programming. It is a structured procedure, utilizing certain techniques. We will cover debugging in this book from the first steps one needs to take in order to find out what went wrong. Programming is a series of many small steps and because some of them can go wrong, we need to construct in small pieces and test them as we go along. We also document our program. We write extra information (comments) about what the instructions do. In order to find where things went wrong, we add extra instructions that give signals of what happens when the program is running. In programming you have complete control over your program and success depends entirely on you. You can make it work. It is you and the computer that gives you feedback. It is a fun procedure like the game hot-cold that children play when they hide something, and another child tries to find it. Immediate feedback is a blessing because you continue your efforts until you succeed. However, this is what causes programmers to work long hours trying to get the program to work. Although programs execute very fast, the first time we run them, we execute them step-by-step, recording the result of every action.

Seeking the Treasure T3

Pic. 14 Scratch map T3



The program in Illustration N16 is written in Scratch and leads the treasure seeker to Treasure T3. The instructions are separated from one another so that the program is not executed all together. Instead, by clicking on each instruction, that one and only that one is executed. This way you can easily spot if any part of the program is not executing as expected. In the above example, this method is suitable, but in other more complicated occasions, other methods will be used that we will cover later. In the map in illustration N15, the cyan circles show the point where the treasurer will be after the execution of each instruction.

Watch video ST3 Scratch T3 step by step for debugging on YouTube

https://youtu.be/rKPQ2A5KNMk

Pic. 15 Scratch T3 Code

when clicked
Move 1 cell Up ^
repeat until touching rope bridge ?
Move 1 cell Right ---->
repeat until not touching rope bridge ?
Move 1 cell Right ---->
Move 1 cell Down V
repeat until touching tree ?
Move 1 cell Right ---->
repeat until touching house ?
Move 1 cell Down V

4 Uncertainty or luck, Elements of chance

(Stochastic games where uncertainty plays a role, as opposed to Deterministic games where there is no uncertainty)

The lack of uncertainty was the reason that the first version of Namatreasure was not a success. I believed at that time, that games of luck do not promote the right values and that people should depend entirely on their own strengths and efforts. It was a lack of knowledge on my part and a disregard of nature's fundamental laws. In nature there is the element of luck, but also the element of change. No two apples are the same; the weather is unpredictable. Even primitive people's survival depended on what they happened to find along their way, a lion, a tiger, or a deer. **The element of luck must be included in the game, and through knowledge and strategy its importance should be minimized.** The element of luck simulates life because life is full of unexpected things. People get very excited by uncertainty, and about lucky games. Consequently, in Namatreasure, we added event cards which the treasure owner writes, and which are associated with locations on the island. When the treasure seeker passes through that specific location, he rolls the dice and depending on the outcome the card may or may not be activated.

The events can be positive or negative to the treasure hunter. In Scratch you get to program uncertain things to happen by using a special command called '**pick random** [1] to [6]' (this outputs a number from 1 to 6 in a random order, like rolling a dice).

Example of Uncertainty in Scratch

Let's investigate an example that will demonstrate the uncertainty element and the commands used to create it.

The project T3V4 is in Scratch in the link

https://scratch.mit.edu/projects/93609347/

(note that the number of the project in the above case it is 93609347). Copy and paste it in any browser to see its operation.

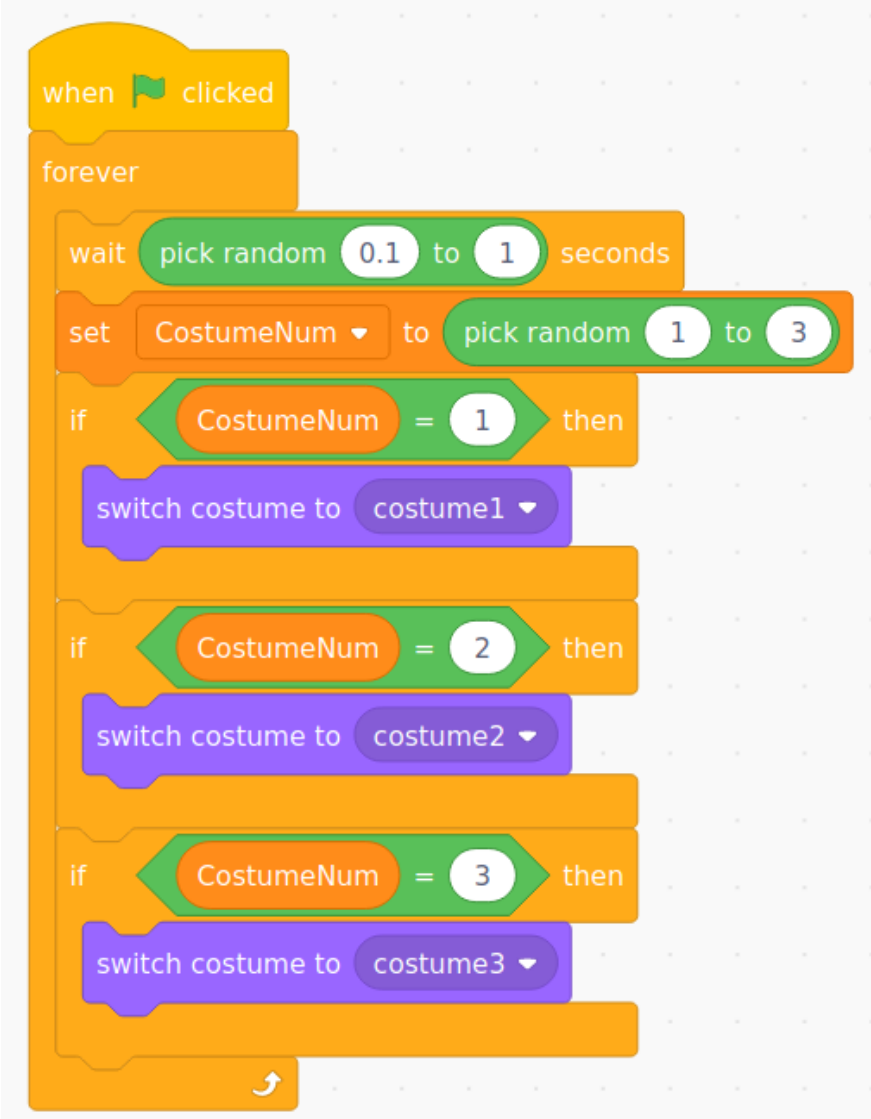
In a game there are lots of things happening, apart from the main character. For example, there may be a flag moving with the wind, waves on the sea breaking on the shore, or snow falling. All of these natural phenomena are stochastic. They happen at random times, in random orders of appearance and intensity, as well as according to other properties. In the program in illustration N13, a flag is programmed to change appearance so that its behavior simulates when there is wind blowing. The simulation is not very convincing or attractive because it changes costume (appearance) every second. Costume 1 shows for one second, then costume 2 shows for one second and then this 'repeats forever'. It looks more like traffic lights than a flag. Its behavior is characterized by its property of "time taken to change costume" (wait for one second).

To create a more realistic looking flag, a new program (in illustration N14) has been written that adds two elements of luck in the behavior of the flag. The program is quite a bit bigger than the previous one, but there is a lot of repetition. The wait instruction has been modified and instead of taking a number value next to the word '**wait**' it now takes a green bar that writes '**pick random** [0.1] **to** [1]' and it does exactly what it says. The green bar '**pick random** [0.1] **to** [1]' is like throwing dice and outputs a random number from 0.1 to 1, including all decimal numbers. It can output numbers like 0.234, 0.996 or 0.567 but not 1.1 or 0.05 or any numbers outside its limits. The number that it generates is written as a value next to '**wait**' and has the same effect as if that number was written there by us. In scholarly terms the value next to '**wait**' that determines its behavior is called a parameter. The command '**pick random** [0.1] **to** [1]' in scholarly terms, is a function (a subprogram) and it outputs a number so that it can be used wherever a number is used. We encountered this type of subprogram in Namatreasure where we used an additional block of commands. However, in Namatreasure, the additional block of commands did not return anything back, but it will in the next chapters, when we revisit Namatreasure. In Scratch, when one element goes into a location and it connects there, it means that it is correct and will function as expected. Therefore, we will continue with intuition, with what seems logical and what we see in the graphical environment, as opposed to mentioning scholarly terms.

Apart from the randomness in '**wait**', there is also randomness in the order that the costumes change. Every costume has a unique number in the upper left corner. The instruction: '**set** [CostumeNum]

to {**pick random** [1] **to** [3]}' gives a value to the variable [CostumeNum].

Pic. 16 Program for flag with uncertainty


when clicked
forever
wait pick random 0.1 to 1 seconds
set CostumeNum to pick random 1 to 3
if CostumeNum = 1 then
switch costume to costume1
if CostumeNum = 2 then
switch costume to costume2
if CostumeNum = 3 then
switch costume to costume3

The [CostumeNum] variable is created before using the button [Make a Variable] from instruction type ‘Data’. After the [CostumeNum] variable gets a number 1 or 2 or 3, it is used for the next three instructions of ‘**If**’. The ‘**If**’ statement is used very often in everyday life and in programming, it is probably the most widely used instruction type. This is the instruction that gives computers and programmable devices the ability to decide based on data stored, or data from sensory devices. Using our experience and intelligence, we analogously program, and by programming we learn lot about ourselves and how we think. Let us continue the execution of the program to the first ’**If**’ statement.

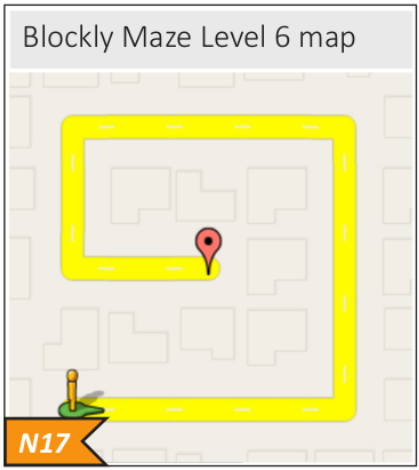
If [CostumeNum] gets the value of 1, then the check inside the green check box (hexagon) is evaluated as true and subsequently, the purple instruction inside it is executed, and thus the flag changes to costume 1. The check in the second ‘**If**’ statement is false because [CostumeNum] is not 2 and the instruction inside it is not executed. The third ‘**If**’ statement happens in a similar manner. The program continues again from the first instruction inside the loop and it continues.

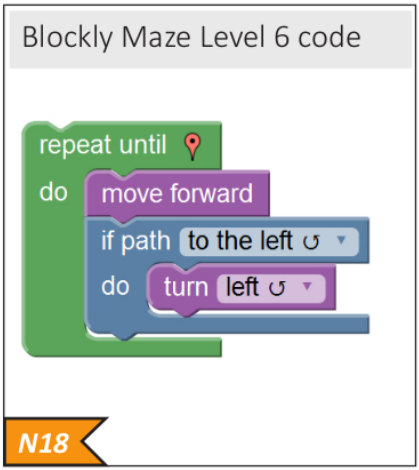
- To investigate the behavior of the program, you can proceed to deconstruct it as in the program of T3 and run each instruction. First press ‘**set** [CostumeNum] **to** {**pick random** [1] **to** [3]}’ and watch the value [CostumeNum] change, although sometimes it does not change because it can return the same value as the previous value. Then press on the ‘**If**’ statements and see how the flag changes when the check is true and how nothing happens when the check is false.

It is worth noting that the program in N14 can be rewritten using two commands inside a loop.

Watch video Scratch Namatreasure ST3 Flag for uncertainty and IF on YouTube https://youtu.be/_XXApVmgXBw

After learning the ‘**If**’ structure, our level is raised and therefore we are ready to try Level 6 of Blockly Maze.





The program to guide the traveler to the red flag can have a maximum of 5 blocks, which we call instructions. Therefore, because there are lots of steps, a repeat until command will be used. Because at some steps there has to be a turn to the left, and because we cannot specify exactly at what point that will happen, we will need to use an **If** statement.

Illustration N18 shows the solution, which is obvious to anyone who understands the type of instructions '**Repeat** u**ntil** (condition)....' and '**If** (condition) t**hen** ...'

We can also continue to Level 7, which builds on Level 6. (Watch video BM6 Blockly Maze Level 6 https://youtu.be/w43NO1op38U)

Watch video BM7 Blockly Maze Level7
https://youtu.be/BfE8PmMMfP0

There are a lot of choices in this map and we need a strategy to reach the target. We need to think first, and then write down the best route. However, when we study it, we reach the conclusion that is exactly the same as Level 6, but this time we move to the right.

Watch video BM8 Blockly Maze Level 8
https://youtu.be/oKwL125BpLY

It builds on Level 6 and Level 7 and again it needs thought.

Watch video BB1 to 3 Blockly Bird Level 1 to 3
https://youtu.be/2SK_uazV82I

Blockly Bird has 10 levels testing your understanding and mastery of the '**If'** statement. The first three levels are within reach of anyone who has attained Blockly Maze 6 and 7.

5 Share it with family and friends, publish it

This is the easiest rule to explain. Before trying to teach my daughters programming, I had already taught them other small things.

But when I tried to teach them programming, they told me "Don't even think about it!". You see I made the mistake of telling them that I was going to teach them programming, rather than telling them that we were going to play. However, it was easy to persuade the parents of their friends that programming is very useful and that I could teach their children, together with my daughters, for free. Of course, my daughters were glad to learn in this context, because they liked the company of their friends. The group was formed with three 10-year-old girls. The lessons had no specific target and the pace was slow, so we all had a lot of fun, and in a year, we managed to build three programs in an environment called "AgentSheets". In the last session, they used their programs to compete with each other. We had three mini competitions, one for each program they had developed. In the end, each one of them won in one of the three competitions which made everyone happy.

Scratch goes very far to make sure there is sharing. First you do not have to download the game, but rather you play it on the web so that you keep your programs online. It also always reminds you to make your project public, to share it with other users. Moreover, it is easy to take a public project and remix it, making it your own, changing it and then sharing it again. In Scratch, they do all they can to promote the sharing of games. However, programming, like any other mental process, requires a lot of working alone.

Investigate the concept of program remixing by seeing inside T3,
https://scratch.mit.edu/projects/93609347/

then selecting [Remix] and changing the route and the cell where the treasure is.

First in order to hide the treasure, you should select the ‘Gift’ sprite and then select the ‘Code’ tab. In the code select instruction ‘**Go to cell** X[ ] Y [ ]’ and change the value of X and Y. Then go to the ‘Mark’ sprite and the ‘Code’ tab. In instructions ‘**Go to cell** X[ ] Y [ ]’ change the values of X and Y so that new marks will be shown to mark the path the treasure seeker will follow. Then go to sprite ‘Level3’ and write the program that will lead it to the treasure.

Some details of Scratch sharing: Once you register, you will get an account with multiple pages. One of the pages is called ‘My Stuff’, where you save all your projects. When you create a new project (select ‘Create’ tab), you will be directed to the editor and in the upper bar there is a magic button called [Share]. If you press it, your project will be shared, as simple as that, with only one click. If someone views your account in ‘**https**://**scratch.mit.edu/youraccountname**’, they will be able to see all your shared projects. They can also see inside your projects and when they are in the edit mode, they can also press the [Remix] button to copy the project into their account so that it becomes theirs. Each project has a tree diagram showing who started it and then who remixed it. In Scratch, there is a Scratch-book for socializing and for finding projects that interest you. There are millions of projects and millions of users, and more are added every day. You can find anything from mathematics and physics simulations, to Mothers’ Day cards. Watch video Scratch Share it with others. https://youtu.be/Lp93CMQ_z5E.

6 Work as a team. Collaboration to attain a task. Engagement

There is a complete chapter dedicated to the important of working in teams. Teams can accomplish more work and better work than one person. Everything we have said above also applies to teams.

To accomplish the above, we use competitions. In order to win you need a team so that more people will contribute to make something bigger and more complicated. Competitions encourage sharing on multiple levels. Firstly, among the team itself, then with the competitors and then with the guides that help in the competition. Competitions also create the need to form teams, to work together and bond and share. They also provide the best awards because the ultimate award is recognition, proving that you are good at something.

In competitions, an element of luck and uncertainty is introduced. The uncertainty is introduced because competitions do not require a fixed solution like in mathematics. Instead, they produce a program that produces different results each time it is run. For example, a competition may be to create a robot that collects oranges. The position of the orange trees is not known, and the robot follows a different route each time and collects a different number of oranges. Effort is rewarded as all participants get a certificate for taking part and some also get an award. There is immediate feedback and failure is not punished.

This leads us to our next chapter.

Congratulations on reaching this level.

You can now start playing with your children using the things you have learned. You have all the basic skills and knowledge to start.

The first thing to do is to make a team of friends. They will spend some time playing programming, but also allow them time to play their own games.

> **MAGIC**: Reward every little bit of effort and never punish failure: your child is a champion. Do not make the mistake that many parents do of expecting your children to know something before they are even taught it, or to expect your children to know something after they are exposed to it only once. The truth is that learning comes with repetition and practice and even if you are a master of something, you can still make a careless mistake. If this lesson, to never punish failure and always reward improvement, is the only thing you take from this book, then you have taken the best bit. Guide children into new knowledge and let them experiment. If they are thinking and doing something, then you have accomplished your purpose. Say “good job” and wait. Good teaching inspires and makes them think in different directions. Let them experiment – this is learning. Tell them that they are good at programming and they will turn the world upside-down to prove that you were right.

KEY CONCEPTS: Use the techniques in computer games to attract the attention of your children. First there is a need to measure progress and to use tangible proof of accomplishments. We have used different thematics with achievement levels in each thematic. For each achievement level your children reach, give them a sticker and a book of achievements to put it in. Use feedback techniques and debugging to test programs during execution. Use elements of uncertainty to create unpredictability and surprises so that they learn to cope in unknown environments. Invite their friends to share learning, post the projects on the web to share them and work in teams. Have them work on a team project, where each team member makes one part.

CHAPTER 5

5 Competitions are a powerful driver.

- Children want to compete
- Children want to become heroes for a day
- Children want to win competitions
- Children take competitions very seriously

1 Children want to compete and win

Children want to compete. We live in a very competitive world and competing is something that must be learned. Also, children like to compete, and they always compare themselves with their peers. Therefore, we must build an environment which includes competitions. I learned about the importance of competitions from a summer camp leader called Father Michael.

I went to Father Michael and told him that I wanted to introduce the children to a game that helps them learn programming – the Namatreasure game. He told me the children do not have time for anything else during their stay at the camp, because all of their time is filled up with competitions. This was completely new to me and amazing because I had never thought about using competitions as a tool to teach programming. So, I went back and tested learning with competitions in a programming summer camp that I had organized. During their stay, the children learned three programming environments and had one competition for each environment.

The first module at the summer school was with Namatreasure, the second with LEGO Mindstorsms and the third with Crouzet, an industrial computer. Children showed equal interest in all the three modules and focused intensely on the competitions and on winning. The subject matter taught was of less importance compared to the importance of the competition and of winning. Children want to prove themselves every day. Children want to compete because they want

to win, which provides proof of accomplishment. It doesn't matter if they play a game and lose 9 out of 10 times. They just want to prove themselves and get better. Every day is a new opportunity for children to prove themselves. They go to school to not only learn, but also to prove themselves, to become a hero for a day. If you ask your children what is important about school, they will all tell you how the teacher favors and praises one child and is unfair to them because he or she doesn't praise them. Very seldom will they talk about something they have learned which impressed them. Every day there must be something to be proud of. Every day you have to accomplish something either by competing or just by doing something.

A competition is good because it sharpens the mind and the comprehension. Even if you lose sometimes, you still win by becoming better and better, and one victory is enough. Programming is like climbing a ladder. It is easy, but you have to take it one step at a time.

There are many failures in learning programming, especially when it is done in the classroom environment. The main reasons for the failures are firstly the teacher and secondly the teaching notes. The teacher is responsible when he or she does not love programming or does not know the subject thoroughly. Most of the time, the teaching notes skip material, do not provide for enough practice and expect children to know something they were not taught.

2 One step at a time

Take one step at a time. This is extremely important, and the steps should be very small. So small that you don't even notice that something new has been introduced. Programming is very easy in small steps. Take for example searching for a treasure. Each step is very easy to be completed correctly. However, there are many steps and if you miss one, you are risking your chances of finding the treasure. So, there are many things which are simple and easy, but they must all be correctly completed. This is also true with mathematics, but you will hardly ever notice it. Solving a mathematical problem is actually solving many small simple problems, but each must be solved one after the other.

3 Competitions in Namatreasure

In the Seeking the Namatreasure tabletop game, the competitions are easy but there have to be two teams. Each team writes instructions for finding a treasure; it specifies the initial cell from where the seeker starts to execute the instructions as well as the cell where the treasure is hidden. They also mark the path that the seeker will follow only on their copy, they do not give it to the other team. Then the teams exchange instructions for the treasures they have hidden. They give only the instructions and the initial start cell. The two teams then start together and have two guesses to find the cell where the treasure is hidden. The team to find the treasure first wins.

Another competition is to give them a path and a story and ask them to write a treasure program, and the first to finish it correctly wins.

4 Competition in Scratch – The Collector

In Scratch, the competitions can be made against any two users so that you can create a championship. For example, in the competition “Robot Orange Collector”, two users can each create a robot that can be loaded to a third account which will act as the playing field. In Scratch, one can load any project that is public and then remix it. After remixing it he or she can load the sprites in the backpack and to another project.

The purpose of the competition is to create a robot that will travel in the field, find an orange, pick it up, take it to the basket, leave it and return to the field for more.

In ‘Code’ tab of ‘robotu’ sprite there are various programs for each level described below and some supporting functions and commands specifically made by us for the project:

Orange pick from tree

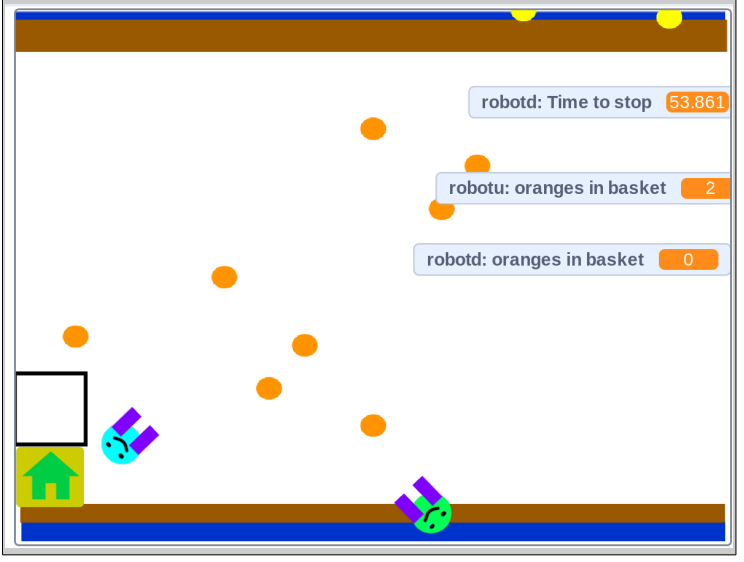
Orange put to basket

Move forward

Leave a trace

The above commands, that can be found at “More Blocks”, simplify some operations and make the program more readable. In any other language, the code for this would be hidden from the user, but in Scratch it cannot be hidden, it can only be moved further down so it won’t be visible at first. You can view the code for these commands, and experiment with it. The best way to solve a problem is to break it down into simpler problems, solve the simple problems and then combine the different solutions.

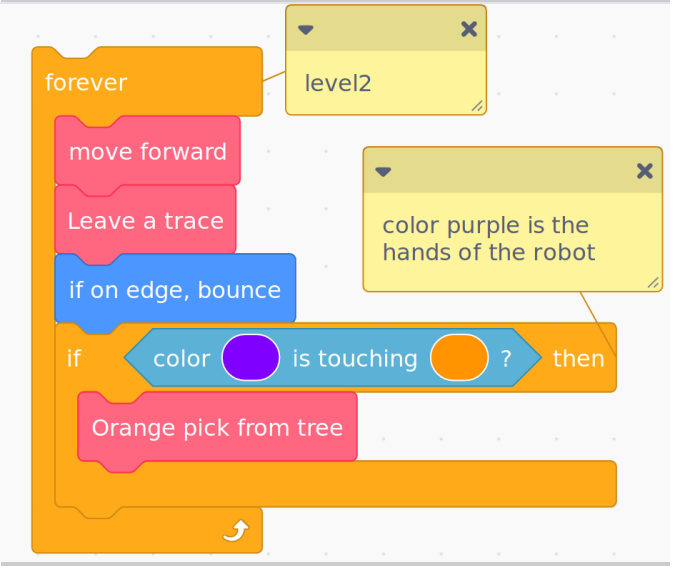
Pic. 19 Stage for the Robot Orange Collector



Level 1: Make the robot move randomly in the field so that it will be able to find oranges. The instruction ‘**leave** a **trace’** can be used to mark the path on which the robot is moving, so that the user can inspect if the robot covers the whole area. The command ‘**pen down**’ can also be used or better results. To access pen commands, press the cyan button in the lower left-hand corner ‘Add extension’ and press ‘Pen raw with sprites’.

Level 2: The solution of Level 1, plus it should be able to pick an orange.

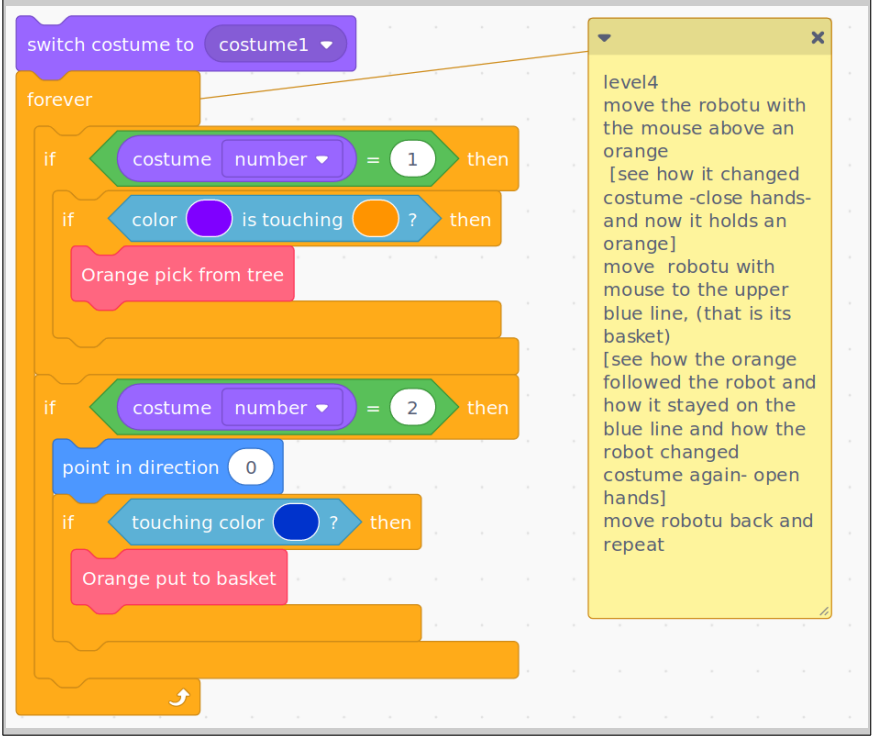
Pic. 20 Level 2 of Robot Orange collector



Level 3: The solution of Level 2, plus it should go to the basket.

Level 4: The robot should be able to pick an orange and put it in the basket. The robot has two distinct phases of operation. In the first phase it is not holding an orange and is searching for one. In the second phase it is holding an orange and will go to the basket to leave it. The different behavior at different phases is very common in video games.

Pic.21 Level 4 of Robot Orange Collector


switch costume to costume1
forever
if costume number = 1 then
if color is touching ? then
Orange pick from tree
if costume number = 2 then
point in direction 0
if touching color ? then
Orange put to basket
level4
move the robotu with the mouse above an orange
[see how it changed costume -close hands- and now it holds an orange]
move robotu with mouse to the upper blue line, (that is its basket)
[see how the orange followed the robot and how it stayed on the blue line and how the robot changed costume again- open hands]
move robotu back and repeat

Hint: Remove the robot's ability to move by itself and use the mouse to move it to investigate its behavior. Later you can add back the ability to move.

Level 5: Add a counter to add the oranges in the basket.

Go to this location to practice with “robot collect competition two robot” https://scratch.mit.edu/projects/93545567/#editor

Add text on how to make a robot to collect oranges.

Watch video SRC1 Scratch Robot collect Orange Pick Level 1
https://youtu.be/taaMY8zq1kQ
Watch video SRC2 Scratch Robot collect Orange Pick Level 2
https://youtu.be/ZDCRnuLSNUE

MAGIC: The subject matter taught is of less importance than the element of competition and of winning. Competitions are a huge driving force that should be used with caution.

KEY CONCEPTS: Use competitions to motivate children. Set the rules from the beginning, make them measurable, objective, easy to enforce and check and execute them in a fair way for all. In Namatreasure there can be competitions between teams. Scratch has robot competitions and Blockly Games is full of tests.

CHAPTER 6

6 Storytelling

- Children like to talk
- Children like to create
- Children want your attention
- Storytelling is an attention magnet
- Storytelling is also one of the greatest skills
- Storytelling keeps your audience's attention

Children like to talk; they like to tell stories and they like to imagine things. So, the next logical step in this simple instruction about programming, should be carried out with a story. As we have used the treasure hunt example before, we will continue with this. It will become even more interesting if there is a story about a witch, a giant, or other mythical creatures.

1 Creating with their imagination

Create a story of going somewhere and finding an Indian chief who gives you a key which you use to open a locker, where you find a sword and food that you take with you. You use the sword to kill the dragon or you use the food to feed the dragon and you make choices which compose an entire story. Depending on where you want to take your hero, you can end it by using the sword or end it by using the food. When the children are in charge of the story, it is much more interesting to them. For example, in Namatreasure you can look at treasure T6. It has been written by a 14-year-old boy with a lot of imagination. In Scratch, there is nothing to encourage this kind of programming, but nothing to discourage it either. In Scratch we can program a treasure hunt that we also wrote in Namatreasure. But in

Scratch, we can add unexpected and hidden surprises. ALICE – a 3D programming environment – has a special edition for storytelling. Children like to create with their imagination. They need to create something, and a story is an exciting creation, especially when it can be shared with friends. It has a start, an end and a body. However, they won't create something if it takes a painful amount of effort or takes a lot of time, unless there is an important reason. This is why we talked earlier about competitions. Because children like stories and creating something and because they like competing and winning, they will be willing to write a long and complicated story so that they can win a competition in Namatreasure. In Namatreasure the two competing teams write instructions for a treasure hunt and then swap their instructions. The winning team is the one that finds the treasure first. A complicated story will take the opposing team longer to find the treasure.

2 Storytelling is an attention magnet

Storytelling is an attention magnet. As the children start telling a story and manage to draw your attention, they will want to tell more stories to get more of your attention because they love to draw the attention of their parents. You can see this with your own children. They always fight to see who will get more of your attention. Storytelling is also one of the greatest marketing skills. If you want to sell something, it must have a convincing story behind it. For example, these shoes are the shoes of NASA astronauts. This food is what astronauts eat for breakfast. Just watch advertisements to verify this. By telling stories, children also learn other skills, such as marketing. Storytelling is never boring. Storytelling draws your attention and makes you visualize things. It's a step-by-step process full of details that draws your attention. It's like the ornament of a drawing or the animation of a movie. People heard their very first stories from their parents when they would go to sleep. They love to feel their parents by their side at night until they go to sleep, so they also love the story. The two things are blended together in our mind. We like storytelling because it reminds us of when we were children. Storytelling becomes part of our game and part of our learning.

Create a program or treasure hunt with adventures, where the seeker tries to find the treasure and also finds enemies that he has to defeat.

4 Treasure T6 (Level 6), a long story

Treasure 6 adds new dimensions to the treasure hunt. It adds the element of storytelling and the element of multiple choices so that the story builds up. This is of the most complex treasure hunt adventures that are featured in Namatreasure, not because it uses complicated structures, but because it is long and has many choices. If you write more complex treasure hunt adventures, we will be glad to include them in the next edition of this book. In treasure T6, you risk not finding the treasure because of your choices, which are based on luck and intuition. From the point of view of the treasure owner who writes the instructions, extra effort is needed to keep track of all the different instruction paths and of what the treasure seeker has in his possession at any time. The story is an attraction magnet that engages the players. It makes the treasure hunt look like an action game. Enjoy the treasure hunt. Please note that the sub block of instructions are not in order, so it becomes difficult to make decisions based on the content of the other sub block instructions you can see.

Watch video T6 Namatreasure Story telling

https://youtu.be/paI2rUzFN_k

Main block of instructions that will lead you to the treasure (Program)			
Treasure Name	T6		
Treasure Island	Island of two rivers		
Initial position in X (column)	4		
Initial position in Y (row)	12		
Num.	Instruction	X	Y
1	**Enter** the house	4	12
2	**Follow** the **Block** “Link1” instructions	4	12
3	**Get out** of the house	4	12
4	**Move** left **until** you see a house above	3	12
5	**Move** up **until** the entrance of the house	3	10
6	**Follow** the **Block** “Link2” instructions	3	10
7	**Move** 1 **cell** up	3	9
8	**Information**: You meet an old lady who is hungry and thirsty, and she asks for food from you	3	9
9	**If** you give food **then** you lose one day of food **Follow Block** “A2” instructions		

	else **Follow Block** “Link3” instructions		
10	**Move** up **until** you see a house with yellow roof on your right	3	5
11	**Eat** and **sleep**		
12	**Move** up **until** you see an Indian hut on your right	3	4
13	**Information**: A monster attacks you		
14	**If** you have a sword **then** you fight and injure it **else** You run to save your life and start from the beginning		

15	**If** you are afraid **then**
	you leave
	
	If you kill it **then**
	remember what you have done
	If you leave food **then**
	
	you lose two days of food		

16	**Move** 1 **cell** right to the Indian hut	4	4
17	**Enter** the Indian hut	4	4
18	**Buy** the canoe in the river		
19	**Move** up **until** you reach the canoe	4	2

20	**Get** in the canoe		
21	**Eat** in the canoe		
22	**Sail** down **until** the bridge		
23	**Take** the canoe on the shore to the right	8	6
24	**Pick** up fruits for two days	9	6
25	**Throw** the canoe on the river		
26	**Get** in the canoe		
27	**Sail** down **until** the two rivers meet	10	9
28	**Sail** 3 **cells** down	10	12
29	**Tie** the canoe in the right pier	10	12
30	**Eat** and **sleep**		
31	**If** you do not have ring **then** **go** home without the treasure **end** of Treasure T6 **else** **wear** the ring		
32	**Dive** into the bottom of the sea		
33	**Enter** the cave that you will find		
34	**Dig until** you find the treasure box		
35	**Pick** one thing to do according to what you have done to the monster:		

	If you did nothing to it **then**
	Follow "E1" instructions
	If you killed it **then**
	Follow E2 instructions
	If you left food **then**
	Follow E3 instructions		

36	**End of treasure** T6		

Sub block of instructions for T6 Link3			
Take with you from this block			
Num.	Instruction	X	Y
1	**Information**: Your behavior makes the old lady very angry, so she sends you back home without the treasure		
2	**End of treasure** T6		

Sub block of instructions for T6 [C]			
Take with you from this block sword or food for five days			
Num.	Instruction	X	Y
1	**Information**: The door opens. Inside the house there is a sword and food for five days		
2	**Pick** only one of the two things in the house that you will need the most in your journey		
3	**Return to the previous block of instructions**		

Sub block of instructions for T6 Link1			
Take with you from this block			
Num.	Instruction	X	Y
1	**Information**: In the house there is an old lady		
2	**If** you speak to the old lady t**hen** **Follow** Block “B” instructions		
3	**Return to the previous block of instructions**		

Sub block of instructions for T6 E1			
Take with you from this block			
Num.	Instruction	X	Y
1	**Information**: The sword tells you that because you were afraid, you cannot take the treasure with you		
2	**Go** back to your house and your family		
3	**Information**: In your home and in your simple life you will find the treasure that you have been seeking and do not look for it elsewhere again		
4	**Return to the previous block of instructions**		

Sub block of instructions for T6 B			
Take with you from this block Information the right key, food			
Num.	Instruction	X	Y

1	**Information**: The old lady tells you that there is a door that has three keys and you have to press the right key otherwise your choice will be your last one		
2	**Take** what the old lady gives you		
3	**Information**: The old lady gives you food for 3 days and she disappears		
4	**Return to the previous block of instructions**		

Sub block of instructions for T6 [B]			
Take with you from this block			
Num.	Instruction	X	Y
1	**Information**: You see a bear running at you		
2	**Run** to the boat to save your life, abandon the treasure hunt		
3	**End of treasure** T6		

Sub block of instructions for T6 Link2			
Take with you from this block			
Num.	Instruction	X	Y
1	**Information**: The door of the house is locked		

2	**If** you try to open it **then**
	follow Block “A*1” instructions
	else
	Follow Block “B*2” instructions”		

3	**Return to the previous block of instructions**		

Sub block of instructions for T6 E2			
Take with you from this block			
Num.	Instruction	X	Y
1	**Information**: The sword tells you that you are a coward because you kill woodland animals that cannot harm you. Go back without the treasure and never come here again		
2	**Return to the previous block of instructions**		

Sub block of instructions for T6 A*1			
Take with you from this block			
Num.	Instruction	X	Y
1	**Information**: On the door there are three buttons [A], [B], [C]		
2	**If** you press [A] t**hen** Follow Block "[A]" instructions		
3	**If** you press [B] t**hen** **follow** Block "[B]" instructions **Else** **follow** Block "[C]" instructions		
4	**Return to the previous block of instructions**		

Sub block of instructions for T6 [A]			
Take with you from this block			
Num.	Instruction	X	Y

1	**Return to the previous block of instructions**		

Sub block of instructions for T6 B*2			
Take with you from this block			
Num.	Instruction	X	Y
1	**Information**: You see a bear running at you		
2	**Run** to the boat to save your life, abandon the treasure hunt		
3	**End of Treasure** T6		

Sub block of instructions for T6 E3			
Take with you from this block treasure			
Num.	Instruction	X	Y
1	**Information**: The sword tells you: You have found the treasure and the magic sword and from now on it is your duty to travel and restore justice		
2	**Return to the previous block of instructions**		

Sub block of instructions for T6 A2			
Take with you from this block ring			
Num.	Instruction	X	Y

1	**Information**: the old lady takes you back to the house. In the house there is an old man. The old man is very glad that the old lady returned home, and he gives you a ring that allows you to stay under water for a long time.		
2	**Return to the previous block of instructions**		

After going on this long and complicated treasure hunt, you will appreciate a summary and an analysis so that it becomes easy to write such long and complicated treasure hunts as a treasure owner and also to find the treasure as seeker.

Analysis of subprogram calls.

Write in a table all the instructions that instruct the hunter to go find and follow another block of instructions

Start from main program (main block of instructions)

Main program line	Call subpro-gram	Instruc-tion number	Call sub-program	Instruc-tion number	Call subpro - gram	Result
3	**Link1**	2	**B**			Information right button, food for 3 days
7	**Link2**	2	**A*1**	2	**[A]**	nothing
				3	**[B]**	run to save your life-End
				3	[C]	sword for food
7	**Link2**	2	**B*2**			run to save your life-End

9	**A2**					ring helps to swim under water
9	**Link3**					End
42	**E1**					End
42	**E2**					End
42	**E3**					Trea- sure

This is a highly complicated treasure hunt, because the hunter must follow three levels of sub blocks. The first level is called from the main block from where the program always starts, then each subsequent call to follow instructions in another sub block is another level.

In real programming each level adds an abstraction level and makes the program more intelligent. By intelligence we mean that with one command, the program does many things, or we mean that the program manages to function correctly in an unknown world. It copes with whatever different situation it encounters. For example, a low-level intelligence robot can follow instructions to go from point A to point B. A higher-level intelligence robot can follow instructions to go from point A to point B but can also avoid obstacles on the way.

> **MAGIC:** Storytelling is one of the greatest marketing skills. Everything has to have a compelling story behind it. Storytelling can be easily practiced in programming, like in Treasure 6 or in creating adventure games in Scratch.
>
> **KEY CONCEPTS:** Encourage children to use their imagination to build a story, thus cultivating other creative talents. When a project requires different talents it also encourages cooperation since it creates a task for many different characters. Storytelling is an attention magnet.

CHAPTER 7

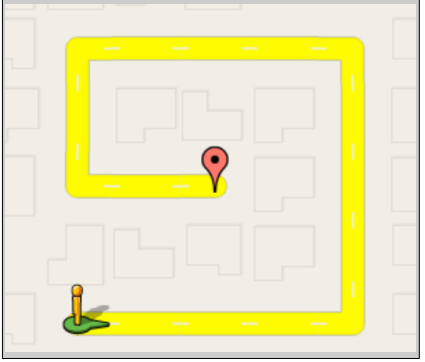
7 Analysis before execution

- First you get confused by the complexity of the problem
- Then you write it all down, separate and analyze
- Then you set priorities
- Then you write the recipe to solve the problem

If you have reached this far it means that you have mastered all the basic programming structures (instruction types, commands, statements etc.).

You are a programmer now, with an understanding of what a program is, what programming language is, how to program in linear mode by storytelling and how to program using a loop where sprites have behaviors and interact with each other.

Pic. 22 Blockly Maze 8 map



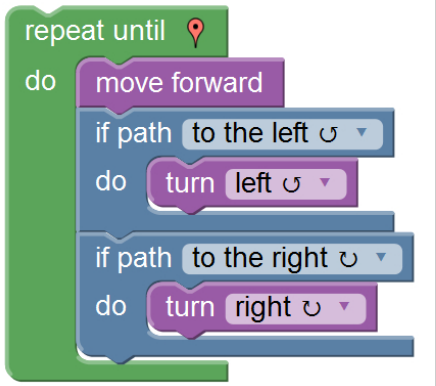
If you understand the previous paragraph, you are ready to go onto a higher level of programming. The problems will be more difficult and will require certain techniques to solve them. We will start with an analysis of the problem, decompose it into simple tasks and then synthesize the different task solutions into one.

Introduce the analysis of the problems Blockly Maze levels 8, 9 and 10. At Maze Level 8 there is an unknown world (the path), but the rules to travel it are easy. You can see the path instantly and you can easily describe it as: Follow the yellow path.

In programming it becomes: if there is a path ahead go forward, if not and there is a path to the left, turn left unless there is a path to the right, then turn right.

Blockly Maze Level 8 is not an advanced level and does not require a high level of analysis. We start from here to have a solid base for the next step.

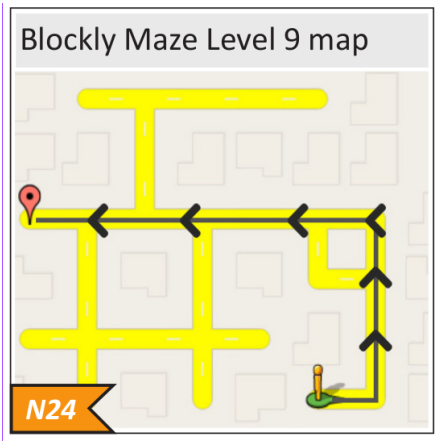
Pic. 23 Blockly Maze 8 code

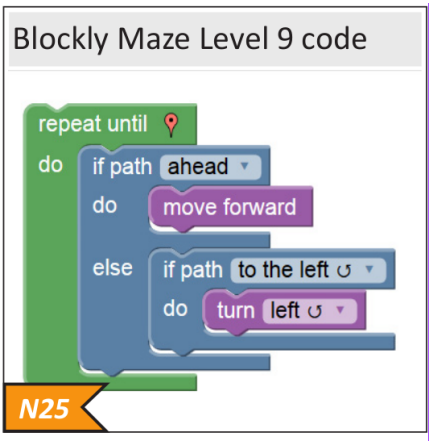


1 Set up priorities

Blockly Maze Level 9:

First, we can draw the path on which the traveler should go. This





is the first part of the analysis. Then we should give the rules that will guide the traveler to this path. If there is a path ahead go forward, else turn left. This is the minimum program to achieve the task. However, we can add more instructions, for example replacing the “turn left” with “if there is path to the left then turn left”. In problem 9 we have set the first priority to

go straight and if it is not possible, turn left. Here is a cascaded **if** statement, where there is an '**If**…' statement within an '**If** ….' statement.

Watch video BM9 Blockly Maze Level 9

https://youtu.be/bt-l8UYw_4E

To visualize these cascaded '**If** ….' statements, think about walking down a corridor where you see a yellow door, a red door and then a green door. You are given the following statements to follow.

```
If you see a green door then
        enter through the door
        If you see a yellow door then
                enter through the door
        else leave the room
else go on
```

When you enter the green door, you see another yellow door and you enter that one. When you open the green door, you are limited to what you see in that room.

Note that you do not enter the yellow door which is in the same corridor as the green door, but you enter the yellow door that is after the green door.

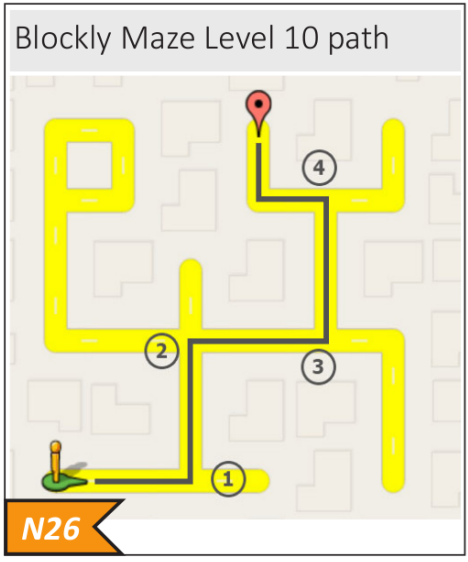
2 Tools for setting priorities

Blockly Maze 10:

Maze Level 10 is extremely difficult and for this reason a hint is provided: "Try following the left-hand wall. Advanced programmers only!"

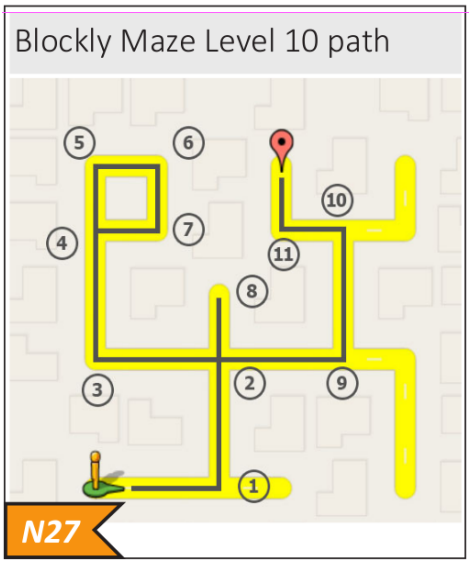
We forget about the message and continue as we have learned. First, we can draw the path that the traveler should go on as it is shown next. Then we see that in locations 1, 2, 3 and 4 there is more than one choice and we have to set the rules for the robot.

	Path left	Path ahead (forward)	Path right	Decide Left, Forward, Right	Priority
1	1	1	0	L	L F
2	1	1	1	R	R L F
3	1	1	0	L	L F
4	1	0	1	L	L R (violation of previous priority



With the available instructions we have, we cannot keep count of the intersections and program what to do at each one. The only thing that can be done is to set priorities like we did in Level 9, to first go straight and if that was not available, to search for an alternative route. In Level 10 we cannot do the same because if we do that then at location 1: the robot will go straight in the wrong direction. From the table above, we see that at intersection 1, left has priority over ahead (forward). From intersection 2, we see that right has priority over the other two, so up to now we have set the priorities as Right, Left, Ahead (forward). From intersection 3, left has priority over ahead, which is as we have it. But at intersection 4, left must have priority over right, which cannot be done because the priorities cannot change in the middle of the game.

After the first failure, we try a different route. Let's remember the hint that was given in the beginning which was "try following the left-hand wall". This means to give priority to left first and see what happens.



	Path left	Path ahead (forward)	Path right	Decide Forward, Right	Priorities 1st, 2nd, 3rd
1	1	1	0	L	L
2	1	1	1	L	L
3	0	0	1		
4	0	1	1	F	L,F first time that we decided that forward has priority over right
5	0	0	1	R	L,F, R priority is set continue to validate
6	0	0	1	R	L,F, R priority is validated
7	0	0	1	R	L,F, R priority is validated

4	1	0	1	L	L,F, R priority is validated
3	1	0		L	L,F, R priority is validated
2	1	1	1	L	L,F, R priority is validated
8	0	0	0	L	When no available route decide to turn Left
8	1	0	0		
2	1	1	1	L	L,F,R validat- ed
9	1	1	0	L	L,F,R validat- ed
10	1	0	1	L	
11	0	0	1	R	

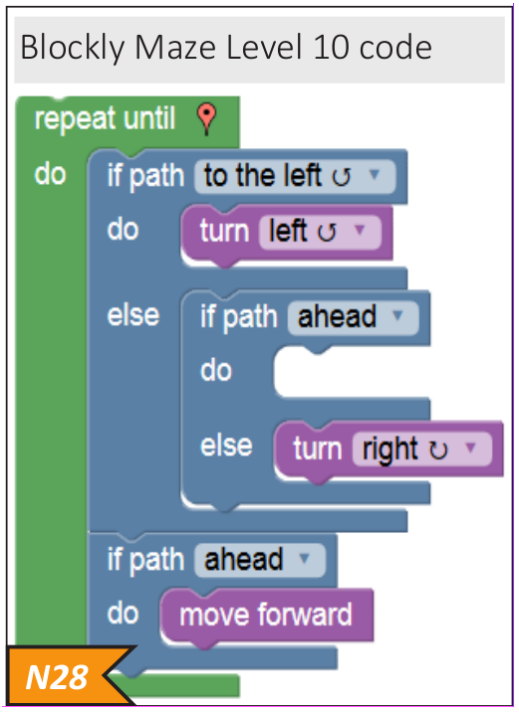
Design a path with priority Left.

Create the table of priorities to test for consistency

First set the priority to left for all the intersections that have left. Then at intersection 4, forward is set at a higher priority than right. The priority left, forward, right, is consistent in all intersections so we can put it in code and test it. If it does not work, we will see where it fails and correct it.

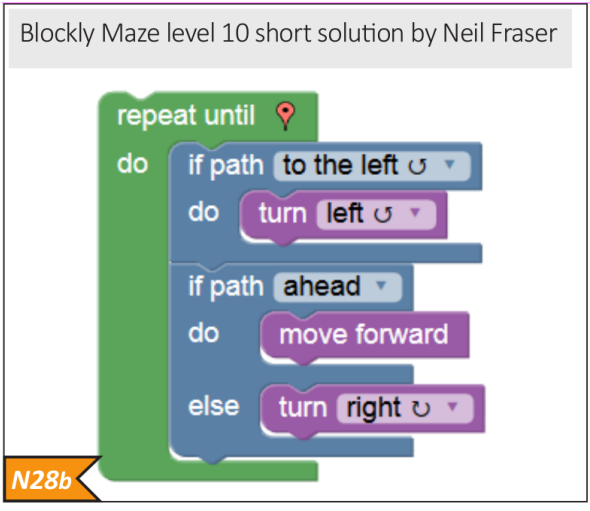
The priorities are shown very clearly, starting from top to bottom inside the first **If**. Priority is left and if there is no left, it is forward, and if there is no left or forward, it is right.

Since all levels have been covered, we have made a similar game in Scratch to practice more.



Watch video BM10 Blockly Maze Level 10

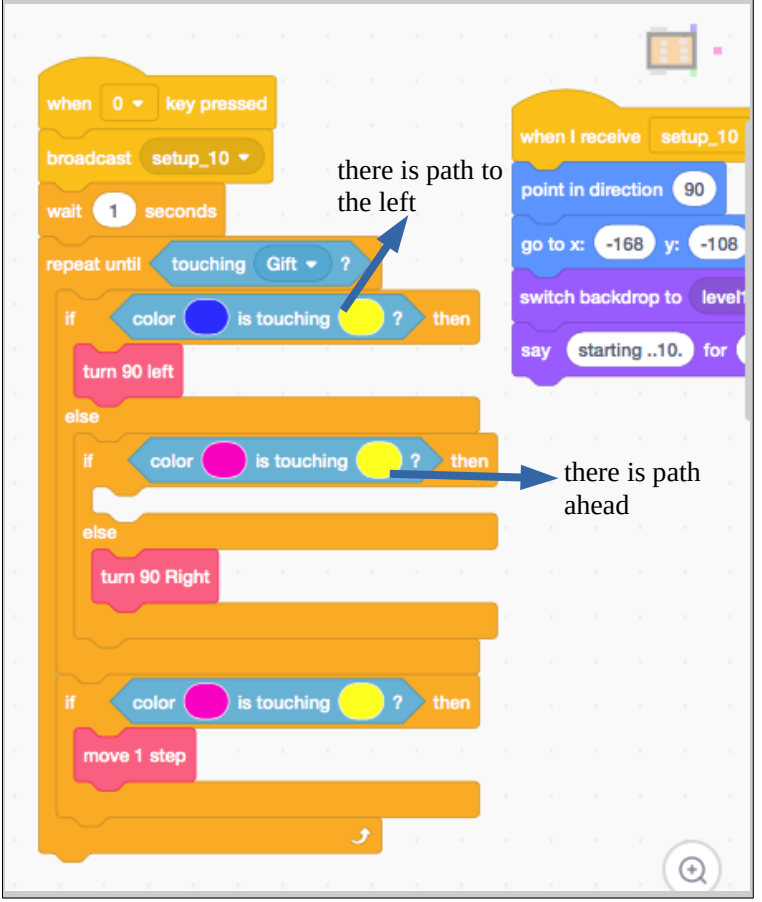
https://youtu.be/wjeUuJUnRZE



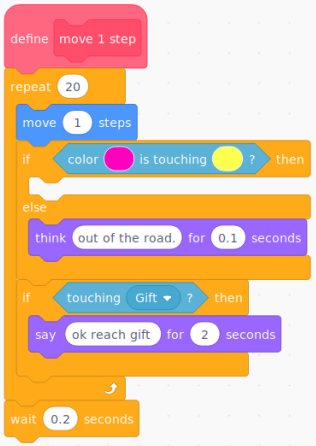
Go to these locations to practice “engino maze level 9 and 10”

https://scratch.mit.edu/projects/98616606/#editor

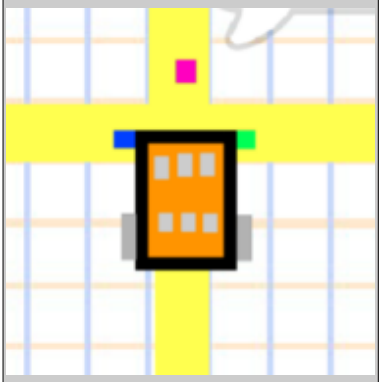
Pic. 29 Scratch project similar to blockly maze 10



Pic 30. Definition of instruction "Move 1 step"


define move 1 step
repeat 20
move 1 steps
if color is touching ? then
else
think out of the road. for 0.1 seconds
if touching Gift ? then
say ok reach gift for 2 seconds
wait 0.2 seconds

Pic 31 Car that moves in the maze, small color rectrangles act as sensors



MAGIC: With programming we can solve difficult problems mainly by breaking the problem into smaller ones using decomposition and analysis. The small problems can be easily solved and tested thoroughly. We can then start the synthesis of the smaller problem solutions, testing every time we add a new module to the project. It is similar to cooking, where the ingredients are added one by one, stirred together, tested and then we proceed to add another one until we have included all of them.

KEY CONCEPTS: Programming is easy but there are difficult problems to solve. Difficult problems face the scientist in NASA and CERN too, therefore do not get discouraged when you face difficult problems. But do not identify difficult problems with programming itself; to solve difficult problems various analytical tools are used.

CHAPTER 8

8 Teams produce more work

- One is not enough
- Two are better than one
- Three can start a team
- Teams can take you to the sky or to zero
- Teams helped primitive people hunt big animals

1 Work in teams

Teams are indispensable. Our whole society is based on teamwork. If you want to accomplish something, you have to be able to work in a team. However, in order to work in teams, you must also learn how to work on your own. Therefore, having fulfilled the previous chapters, we have reached the point where our children are fluent in programming. They know their limitations, and they can now start to work in teams. There needs to be a motive though. One motive is company. Children like to have company. They like to talk and make jokes. Another motive for working in teams is that together with their team, they can win. For example, if there is a large amount of work to be done and each member of the team does a part of it, then the team can win over an individual. As teamwork is important and necessary and we have reached a point where we can work in teams, let us now concentrate on what the issues are when working as a team.

First, we must be careful. Working in teams has many traps. Children can easily form teams and make friends, but the opposite is also true: children can also easily break up teams and lose friendships. They can easily have fun and can also easily disagree and fight. We must find all of these pitfalls and remove them before the game starts. Each member must feel important to the team and must have a task.

You cannot have two leaders in the same team. You can only have one leader. You will also have one member who is the best follower and the others who work the team. It is better if they know each other a bit.

2 Selection of the team members

When selecting the teams, try to have one leader in each group, as well as one member that bonds the team together, a selection of workers that work on the real stuff, and someone who is a dreamer who can see the big picture and can create lots of ideas. Above all, do not deprive anyone of the right to become a member of the team. Take extra steps to find the child who is a secret team leader – who cannot take orders from anyone but is too shy to take control. It is worth the effort to try to bring everyone on board.

Once the team is formed, you should see them as one entity and talk to them as a whole. In private you can talk to anyone you want, but in public, when you talk to the team, you must speak to them collectively rather than to one individual. I cannot stress the importance of this point enough.

If you decide to give praise to someone, you must find a way to also give praise to each person on the team and distribute the praise and comments to all of them. You must give all of them credit. Although we say all these things about teams, **the most important motive for children to work in teams is that it increases their chances of winning**. And as we already know, it's all about competition and winning. It's all about feeling important. Give the teams a task that they have to separate into smaller tasks so that together as a team, they produce more than they would be able to on their own. This will increase their chances of winning and will make them want to work on teams in the future. They will appreciate that working with teammates increases their chances of winning. They will only appreciate their teammates if their individual work increases the chances of the team winning. If any of the team members do not work, the others will want to remove them from the team.

Children are very straightforward. If someone adds to the team, they will want him or her on their team. If someone subtracts from the

team, they will not want him or her on their team. So, you must make sure that each member contributes to the procedure.

Up until now, we have seen difficult problems that could be easier to solve as a team than as an individual. For example, in Namatreasure, treasure T6, in Blockly Maze Level 10, and in Scratch the orange robot competition.

An excellent game for team work is also the “tennis competition” project, which has an execution part and a programming part. Some members of the team will be better at the execution part and others at writing the code, but the synergy will come when they exchange ideas because both are needed to get the maximum impact. https://scratch.mit.edu/projects/98775866/#editor

The project is the tennis competition remix.

> **MAGIC:** Children’s motive for working in teams is that together with their team, they can win. For example, if there is a large amount of work to be done and each member of the team does a part of it, then the team can win over an individual.

> **KEY CONCEPTS:** Although teams produce more than an individual can, it is more difficult to manage teams. Everyone has an opinion and the need to participate, to be heard and to be the leader. Managing the team and selecting the team members is much more difficult than programming although for some people it comes naturally. These people make the best leaders.

CHAPTER 9

9 To 3D world and beyond

- You are now a programmer
- The road lies ahead and it's up to you where you will travel
- You can do mobile applications with Blockly MIT App Inventor
- You can do automation projects with Blockly and Arduino
- Programming is for artists too

Where do we go from here? We have learned the basics of programming in environments that were designed for learning and now we need to go to the outside world. It is as if we have just got our driving license and are free to drive on any road we want. The journey and how long we travel is up to us.

The last thing we will do in this book is to show you different exciting destinations that are within your reach. With the experience and knowledge you have acquired you are now ready to start learning to use tutorials.

You will be thrilled and amazed with what you can do with the things you have learned. It will come as a huge surprise to you that you can still do a lot with these environments. If you are wondering why there is so much activity around these learning environments, you have to first think about their success. Both Scratch and Blockly are very popular and in the world of the internet the winner takes it all. This is so true, that a new paradigm has been formed. First you provide a free service to gather an audience and then you charge for a better service, exploit their presence on your site or have a gift shop at the exit. The gift shop provides for the free service.

1 Mobile Application development.

Can I program my mobile device? Yes you can program your mobile device and develop applications for it, just like you have written programs for Blockly, using the MIT app inventor.

“MIT Center for Mobile Learning.Welcome to MIT’s Center for Mobile Learning’s App Inventor website (the “Site”). The Site runs on Google’s App Engine service.”

What is “Google’s App Engine”? A quotation from the website: “Google App Engine: Platform as a Service, Google App Engine lets you build and run applications on Google’s infrastructure. App Engine applications are easy to create, easy to maintain and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers for you to maintain. You simply upload your application and it’s ready to go.”

In simple words, imagine that you are a car designer, Google App Engine would give you the tools to design, provide the factory to produce the first few cars, and if sales pick up they will provide the facilities to produce more, to meet the demand without any limitations, and take care of the delivery to the customer’s door. They will only charge a service fee for the exact amount of services you have used. It is an excellent deal that is only possible with software products. The MIT App Inventor does exactly that but for free because we are still evolving, we are not designers yet, we are learners, we are still in an incubator.

Quotation from the MIT app inventor website: MIT app inventor:

“Description of MIT App Inventor

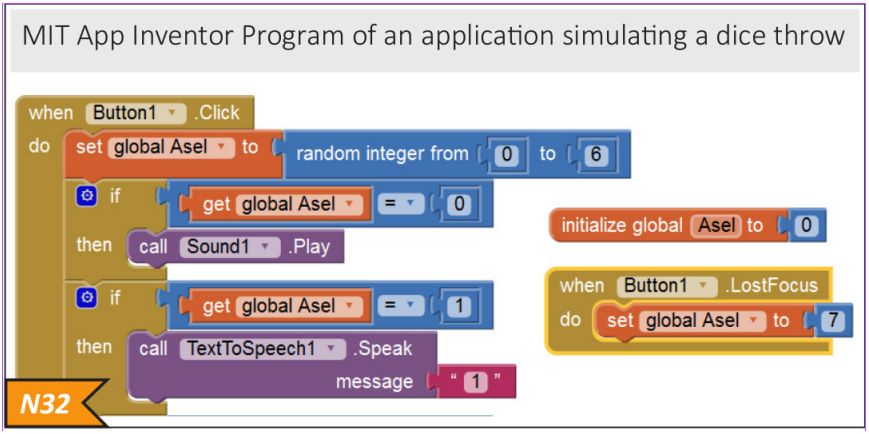
From this Site you can access MIT App Inventor, which lets you develop applications for Android devices using a web browser and either a connected phone or emulator. You can also use the Site to store your work and keep track of your projects. App Inventor was originally developed by Google. The Site also includes documentation and educational content, and this is being licensed to you under the Creative Commons Attribution 4.0 International license (CC BY 4.0).

The Apps you create

By creating and storing apps on the MIT App Inventor server you represent and warrant that you are the owner and creator of the apps, (i) that you have the authority to authorize MIT to store the apps on the MIT App Inventor Server and (ii) you will use the apps in compliance with all applicable laws and regulations. You, and not MIT, are solely responsible for your apps and your use of them.

MIT has no proprietary rights in the apps you create with MIT App Inventor. These apps belong to you. Your apps are stored on the MIT App Inventor server."

So, we can become developers for free, in the real world, and do whatever we want with our applications. For example, you can write one to remind your husband of the things you want him to do.


MIT App Inventor Program of an application simulating a dice throw
when Button1 .Click
do set global Asel to random integer from 0 to 6
if get global Asel = 0
then call Sound1 .Play
if get global Asel = 1
then call TextToSpeech1 .Speak
message " 1 "
initialize global Asel to 0
when Button1 .LostFocus
do set global Asel to 7
N32

Above you can see a program that is written in MIT Appinventor, but it also looks as if it is a Scratch program. The program refers to a Button 1 on the screen that you click (you tap it). Whenever you click it, it executes the following program. The program executes only when button 1 is clicked. First it selects a number from 0 to 6 and assigns it to a global variable called Asel. All subprograms in the project know this variable. Then there are "If" statements for every possible number. For example, when Asel becomes 1, the Function 'TextToSpeech1' is called and speaks the text in parenthesis, which is 1. When variable Asel becomes 0 it is invalid, and simulates

the throw of dice out of bounds. When Asel becomes 0, it plays the sound with the name Sound1, that can be anything you record.

The rest of the program is not shown but it is made of similar “if” statements as the “if” statement for number 1. Before using a variable, it is initialized using the command “initialize global Asel to 0”.

2 Programming in art

Programming language Processing. (reference https://processing.org)

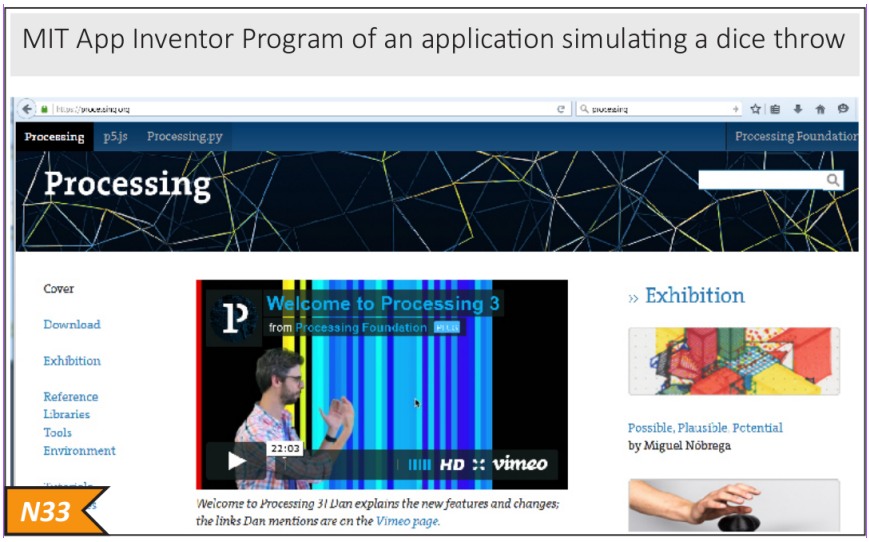
Processing is one of the programming languages which uses the curly brackets. We propose this as the first language with curly brackets that you learn, if you want to expand into more established languages that most programmers use today.

Processing is open, free and it addresses the needs of artists. It is a tool for artists that enables them to create visual arts. Coming from the engineering field, it came as a big surprise to me when I learned that artists use programming, but then again, artists can use everything and anything to create. For artists, programming is just another tool that will help them create.

In this book, we claim that anyone can program. Artists do whatever they do best, therefore it is not a surprise that processing has some of the best tutorials. Do not miss their first tutorial in this link http://hello.processing.org/editor/. Everything is free, but if you want to buy a book on processing please buy it from their store https://processing.org/shop/ so that you can sponsor their activities.

Useful quotations from their site: (2016)

“At our core is the philosophy and politics of open-source software. We see software as a medium, or something that connects two things. We view it as a means for thinking and making. We believe it should be free. We believe that learning to program is not about acquiring a certain skillset but is instead a creative and exploratory process. We believe software, and the tools to learn it, should be accessible to everyone. We believe software literacy and an understanding of media of all kinds is essential knowledge for today.”



“For the past fourteen years, Processing has promoted software literacy,

particularly within the visual arts, and visual literacy within technology. Initially created to serve as a software sketchbook and to teach programming fundamentals within a visual context. Processing has also evolved into a development tool for professionals. The Processing software is free and open source, and runs on the Mac, Windows, and GNU/Linux platforms.”

Education

“From the beginning, Processing was designed as a first programming language. Processing is geared toward creating visual, interactive media, so the first programs start with practicing drawing. Students new to programming find it incredibly satisfying to make something appear on their screen within moments of using the software. This motivating curriculum has proved successful for leading design, art, and architecture students into programming and for engaging the wider student body in general...”

Culture

“The Processing software is used by thousands of visual designers, artists, and architects to create their works. Projects created with Processing have been featured at the Museum of Modern Art in New York, the Victoria and Albert Museum in London, the Centre Georges Pompidou in Paris, and many other prominent venues. Processing is used to create projected stage designs for dance and music performances; to generate images for music videos and film; to export images for posters, magazines, and books; and to create interactive installations in galleries, in museums, and on the street. ….”

History

“Processing was started by Ben Fry and Casey Reas in the spring of 2001, while both were graduate students at the MIT Media Lab within John Maeda’s Aesthetics and Computation research group. Development continued in their free time while Casey pursued his artistic and teaching career and Ben pursued a Ph.D. and founded Fathom Information Design. Many of the ideas in Processing go back to Muriel Cooper’s Visual Language Workshop, and it grew directly out of Maeda’s “Design By Numbers” project, developed at the Media Lab and released in 1999. The Wiring and Arduino projects, in turn, grew out of Processing while Casey was teaching at the Interaction Design Institute Ivrea in Italy.”

3 Programming intelligent devices at home

The previous exciting story leads us to a new even more thrilling one, the Arduino.

Arduino is a major milestone in the maker’s movement. The maker’s movement empowers ordinary people like you and me to become makers of things in the three-dimensional world, home automation, smart devices in agriculture, smart manufacturing, fashion and wearable devices, toys and in any aspect of our life.

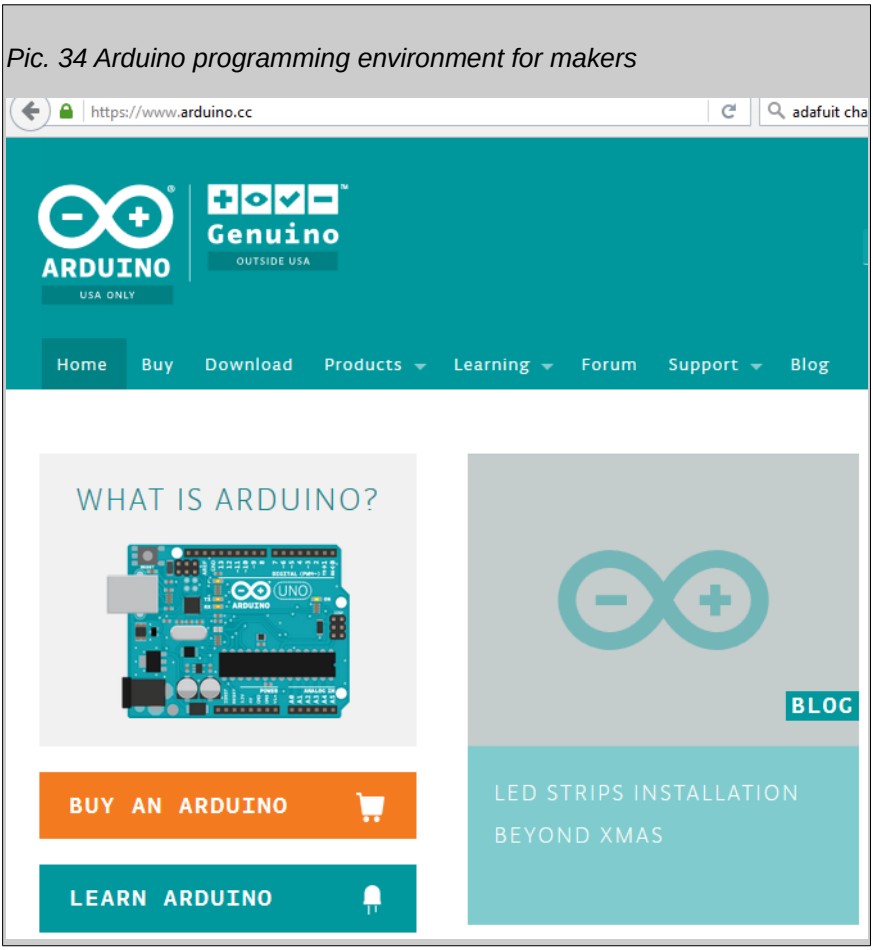
Arduino is a free open platform with thousands of devices to meet any application or need. What ‘open’ and ‘free’ means in this context, is that anyone can copy the design, manufacture it and sell it. This is what happens with Arduino. There is the official Arduino site where you can buy Arduino-certified products, usually development boards.

There are other manufacturers that manufacture authentic Arduino copies and they pay a small fee to the Arduino parent company. Then there are other manufacturers that manufacture the exact Arduino projects but do not put the Arduino mark on them because they did not pay royalties to Arduino. They say their products are compatible with Arduino and give them similar names like Beduino. Because it is open and free to copy, Arduino became a huge success that continually evolves.

From the official website, what is Arduino?

"*Arduino is an open-source prototyping platform based on easy- to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the micro- controller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.*

Pic. 34 Arduino programming environment for makers



Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals has gathered around this open-source platform, and their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

[pic N34 Arduino Programming environment for makers]

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider

community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide." …

Why Arduino

• **Inexpensive** Arduino boards are relatively inexpensive compared to other microcontroller platforms because they are so popular and because any manufacturer can produce them.

• **Cross-platform** The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems, so you can run it on any computer.

• **Simple, clear programming environment** The Arduino Software (IDE) is easy to use for beginners, yet flexible enough for advanced users to take advantage as well. And you can also use the professional environment supplied by the Chip manufacturer that is also free.

• **Open source and extensible software** The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

• **Open source and extensible hardware** The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. You can start any project you want from the closest available and then continue from there. It is very useful to have an initial working prototype.

In simple words, Arduino is good for everyone, from the beginner to the professional, because anyone can find cheap, excellent quality products covering all the available applications. You start in an easy

programming environment and then move to a more complicated environment with many buttons and windows.

Fortunately for us, we can program Arduino in Blockly environments and in Scratch. From Scratch 1, the Snap4Arduino environment has been developed, where projects can connect to the outside world. From Blockly, an environment called ArduinoBlockly has been developed to program Arduino boards.

Snap4Arduino and S4A are Scratch environments that have extra instructions for controlling the Arduino board. Once you develop the program, test it and debug it using the excellent Scratch tools. You can then easily translate it to the Arduino programming language.

> **Magic:** Programming is everywhere in mobile applications, in art, in home devices, on the web, in industry, in medical engineering, and you can also develop your own solutions. You cannot excel in all those fields, but you can excel in one or two, and understand the others. If you understand programming, you can appreciate the potentials and the weaknesses that devices running programs have. Above all, you will appreciate human behavior and understand “IF Then” statements and “While” loops that form the behavior of people. You will see the similarities between our human behavior and that of programmable devices.

Appendix 1. Level and Certificates

The learning path followed in this book is tested and is easy to follow with exciting discoveries and projects along the way. The path can be augmented with more projects and examples that you can add at “Namatreasure Play with Children Programming” Studio in Scratch.

The awards in this Appendix are awarded to learners that repeatedly demonstrate the ability that each award requires. For Blockly Maze one can use the project in the book to add more examples and tests.

Children that manage to collect all the awards are fluent in programming, since they have mastered the foundations and are comfortable in talking about solving programming problems. They can start a project from start to finish, work in teams and talk about project development.

The material covered in this book is essential for learners irrespective of age, the only difference related to age, is that older children can go through the material very fast.

Learners that have mastered all the awards in this Appendix are ready to go on to the next level, where real life problems can be solved using programming and other hardware, which is the subject of our next book. First, we explore the problem by defining it and by enumerating the steps needed to solve it. The exploration continues in the Scratch environment, where the problem is simulated and possible solutions are investigated. After simulation, a test facility is built to test the project in a laboratory environment using S4A, Arduino hardware and software and Engino mechanical parts. A toy test facility is built and tested, while we continue programming in the Scratch environment, and we can control and sense the real world with motors and sensors.

After the project has been thoroughly tested in the test facility, we can proceed to build a real project. The program is moved from Scratch to Arduino and the hardware is built using real hardware items scaled up to its real dimensions.

NAMA
Treasure

CERTIFICATE

This is to certify that

.....

for showing competence and knowledge in achieving full mastery of the following certificates.

.....

.....

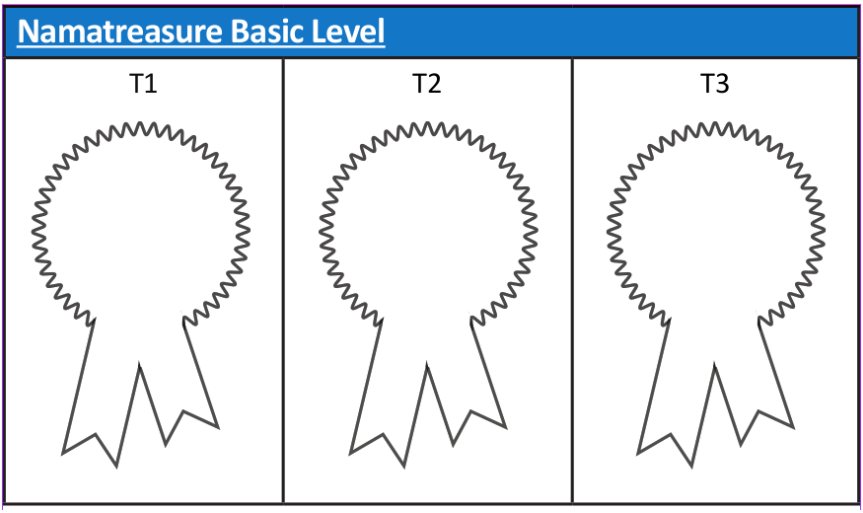
.....

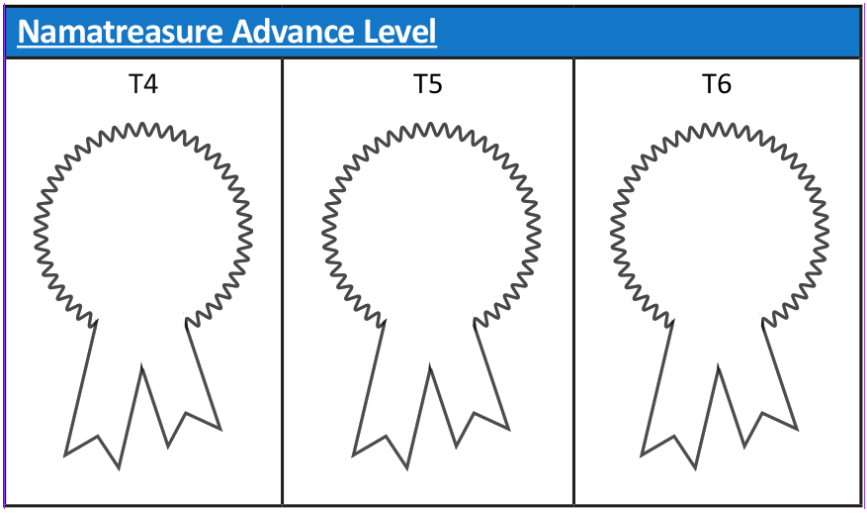
.....

.....

Signature of Examiner

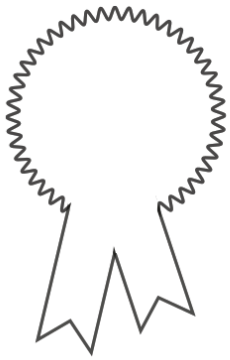
_____


Namatreasure Basic Level
T1
T2
T3

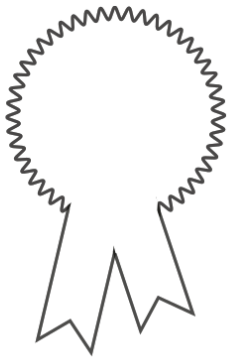

Namatreasure Advance Level
T4
T5
T6

Blockly Maze Basic Level

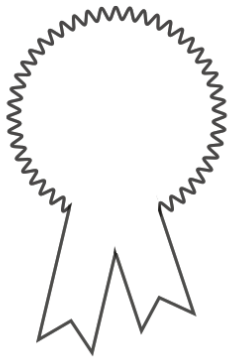
1



2

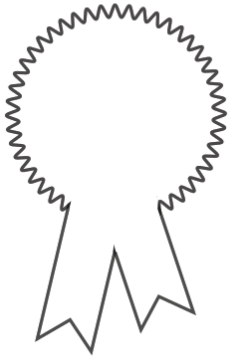


3

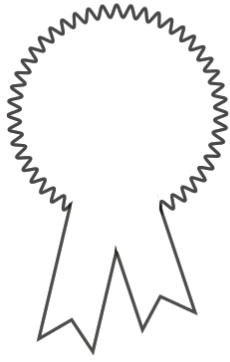


Blockly Maze Intermediate Level

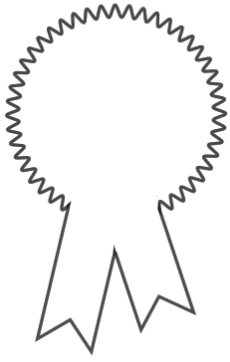
4

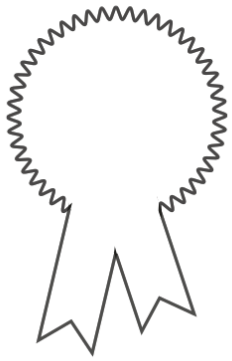
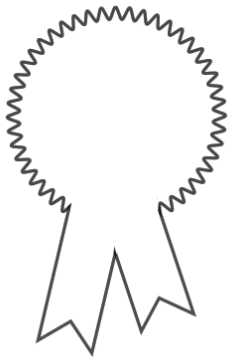
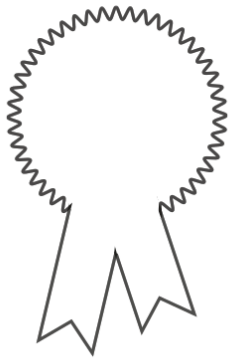


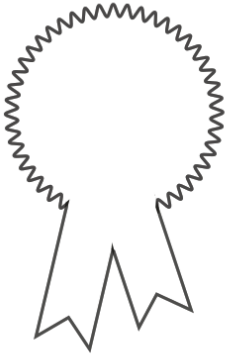
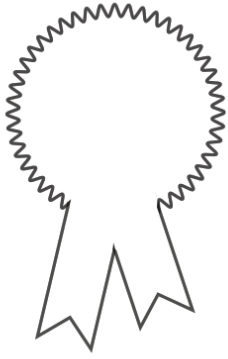
5



6



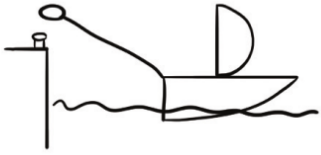
Blockly Maze Advanced Level		
7	8	9
		

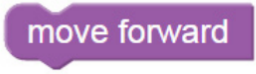
Scratch Robot Collect	
Move Everywhere	Two phases, Collect and Store
	

Appendix 2 Programming instruction

types and control structures

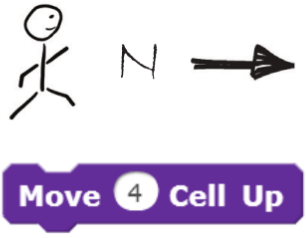
Wherever there is the character ‘N’ it can be replaced with a number. Wherever you see the phrase ‘Nam’ is represents Namatreasure, and similarly ‘Sc’ with Scratch and ‘Bl’ with Blockly.

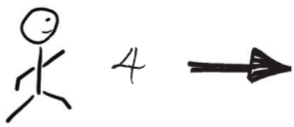
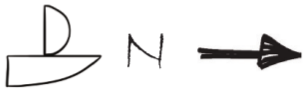
Nam: Tie the boat to shore	
Nam: Enter through the door	
Nam: Enter in the house	

Nam: Move across the bridge	
Bl: Move 1 cell forward	

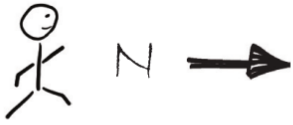
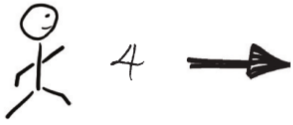
Simple Instruction types with parameters

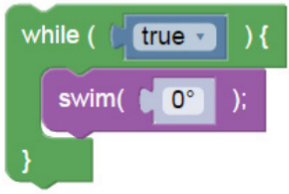
Sc: Wait for one second	

Nam: Move N cells right **Sc:** Move N cell up	
Bl: Turn 90 degrees clockwise Turn 90 degrees anticlockwise (left and right are parameters in a drop down list, list appears when the arrow is pressed)	

Nam: Example Move 4 cells right (N can be replaced with any number. In this case it can not be a fraction) The arrow can be replaced with any arrow pointing to any other direction	
Nam: Sail N cell right	

Simple Loop structures (they are called structures because they are composed of more than one instruction and they contain other instructions. They are also called control structures because they control the flow of the instructions executed.)

Nam: Move N steps right **Sc:** Repeat N times Move left Go back to repeat	
Nam: Move 4 steps right Repeat 4 times Move 1 cell right Go back to repeat	

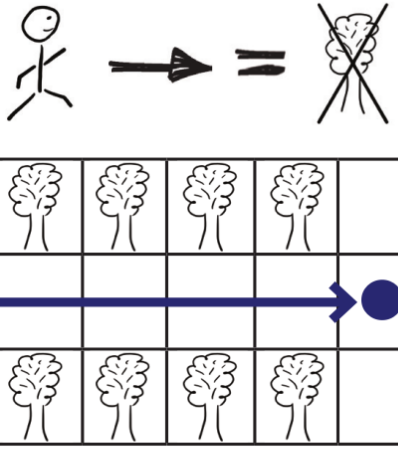
Bl: Repeat forever	
Sc: Repeat forever	

Advanced Loop Instructions with Condition

Nam: Move Right Until..... or **Nam:** Repeat Until..... Move right Go back to Repeat **Bl:** Repeat until question Move N cell right	

Nam:
Move right until there are no trees

Repeat until there are no trees
    Move right
Go back to Repeat

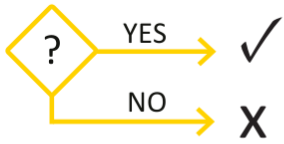


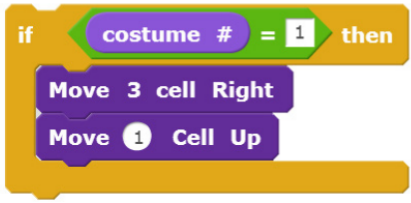
Nam:
If question = true
then .....
else .....

(the question in formal terms is called an expression and the whole statement structure is called conditional because the instructions within this structure are executed under certain conditions)

Sc:
If question then.....

Bl:
If question do.....
else



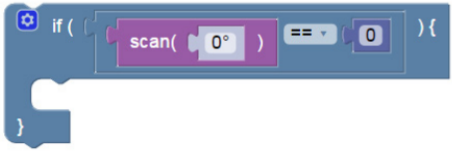


Nam: Example If it rains then follow the yellow instructions else follow the green instructions or you might think of it as Does it rain? If the answer is yes then follow the yellow instructions else follow the green instructions	

Sense instructions:

Nam: Type Move direction until you sense …….. Example Move up until you see bridge on your right **Sc:** Is sprite touching the bridge (bridge is another sprite) **Bl:** Scan, finds the distance from an obstacle at an angle 0 degrees	

Functions (This is advanced and it is not used in Namatreasure) can be used as parameters in other statements

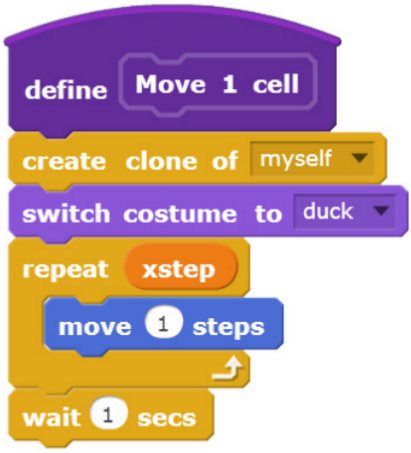
Sc: Pick random is a function that produces a random number from 01. to 1 and that numer goes into the wait instruction as parameter	
Bl: Scan is function that produces the distance of an obstacle at an angle of 0 that goes into a numerical comparison	

Define new instructions

You can define instructions any time with a graphical design and explain it in your language. For all those that do not speak your language, you can hope they understand your design.

Sub programs, calls to run subprograms

Nam: Go to B2 instructions (when they are finished go to the next instruction in the main program)	

Sc:	 define Move 1 cell create clone of myself switch costume to duck repeat xstep move 1 steps wait 1 secs

Appendix 3 Further Reading

References are written in a free form so that you can add your own references later in the same way.

1. TED talk - Let's teach kids to code

Young people today have lots of experience interacting with new technologies, but a lot less so of creating [or] expressing themselves with new technologies. It's almost as if they can read but not write."

Mitch Resnick Let's teach kids to code

www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code

2. Evolution of Graphical programming languages

Source Wikipedia

Agentsheets

Influenced by: Lisp, Logo, Smalltalk

Influenced: Etoys, Scratch

Designed by: Alexander Repenning Developer:

Etoys (Programming language)

Influenced by: Logo, Smalltalk, HyperCard, StarLogo, AgentSheets

Influenced: Scratch

Designed by: Alan Kay

Developer: Scott Wallace, Ted Kaehler, John Maloney, Andreas Raab, Dan Ingalls

Scratch (Programming language)

Influenced by: Logo, Smalltalk, HyperCard, StarLogo, AgentSheets, Etoys

Influenced: Scratch Jr Snap!

Designed by: Mitchel Resnick

Developer: MIT Media Lab Lifelong Kindergarten Group, John Maloney,

Follow the Etoys Programming Language team that will lead you to the MIT scratch project. Follow the Agentsheets story that although first in graphical programming stayed behind in popularity because in my opinion it was not free. Because it was not free it could not be adopted widely by schools and since most other programming languages were free it was not welcomed.

3. Computer science in Vietnam, Neils Blocky (Google)

https://neil.fraser.name/news/2013/03/16/

https://neil.fraser.name/news/2012/01/24/

https://neil.fraser.name/news/2012/06/19/

https://neil.fraser.name/news/2013/12/31/

4. Etoys Alan kay talk on TED

“With all the intensity and brilliance for which he is known, Alan Kay envisions better techniques for teaching kids by using computers to

illustrate experience in ways – mathematically and scientifically — that only computers can.” https://goo.gl/KAbpav

The Etoys system is based on the idea of programmable virtual entities behaving on the computer screen.

5. Tyler DeWitt:

Hey science teachers — make it fun TEDxBeaconStreet 2012 · 11:20 · Filmed Nov 2012 https://goo.gl/yopq0I

6 Blockly showcase of environments that use Blockly

https://developers.google.com/blockly/

7 https://developers.googleblog.com/2019/01/scratch-30s-new-programming-blocks.html

On January 2nd 2019, Scratch 3.0 launched with all of the code open source and publicly developed. At Google, we created two coding activities that showcase this code base. The first was Code a Snowflake, which was used by millions of kids as part of Google's Santa Tracker. The second was a Google Doodle that celebrated 50 years of kids coding and gave millions of people their first experience with block programming.

7.1 https://www.google.com/doodles/celebrating-50-years-of-kids-coding

7.2 https://santatracker.google.com/village.html

Acknowledgments

I wish to acknowledge all the help that my students have given me, helping me to understand what is important and teaching me new ways to improve the material in this book. I also want to thank my family for their support and for the time they have given me to pursue what is important to me.

Many thanks go to Andreas Kyprianou for providing the look and feel of the book, and for making it pleasant to the eye and to Emily Millett and Timothy Isaacs for providing the editing and making it simple to read.

I also acknowledge the USA government, first for the CASP Cyprus American Scholarship Program, and second for the support they give to the development of programming languages for children that are translated in many languages.

About

Nicos Ioannou

As a child, Nicos was a dreamer who liked to create things in his imagination, hoping that one day he would make them a reality. As he got older, he enjoyed learning more and more as the years went by. He became an electronic engineer and worked in a power station for ten years, before going on to work in the public service, where he is engaged in research and innovation.

He has not forgotten the joy of creating and always dreams of finding ways to make learning interesting, enjoyable and creative. He believes that intellectual property is important, therefore he has developed two patents. He also believes that programming is important and therefore has developed useful programmable devices using Arduino. Nicos practices first and then shows others the things he found interesting and useful.

Nicos studied in the United States in Los Angeles USC and the University of Texas, Austin, with scholarships from (CASP) Cyprus America Scholarship Program.